MICROSERVICES IN THREE WEEKS

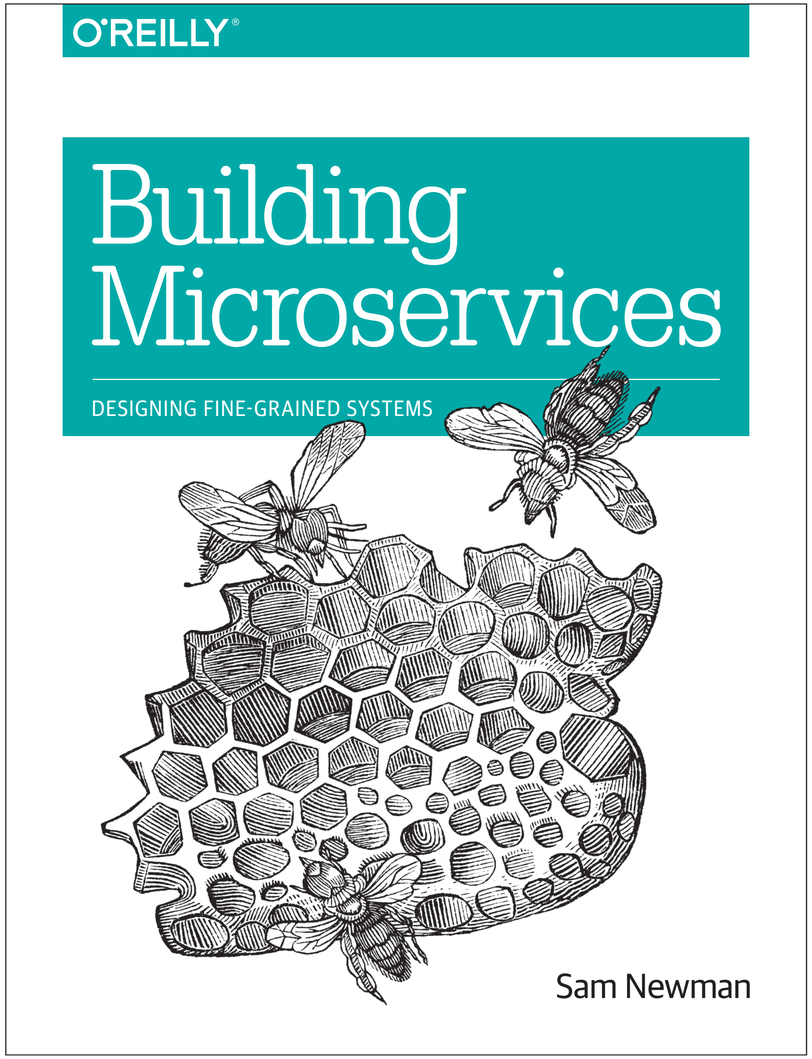
WEEK 1: FUNDAMENTALS

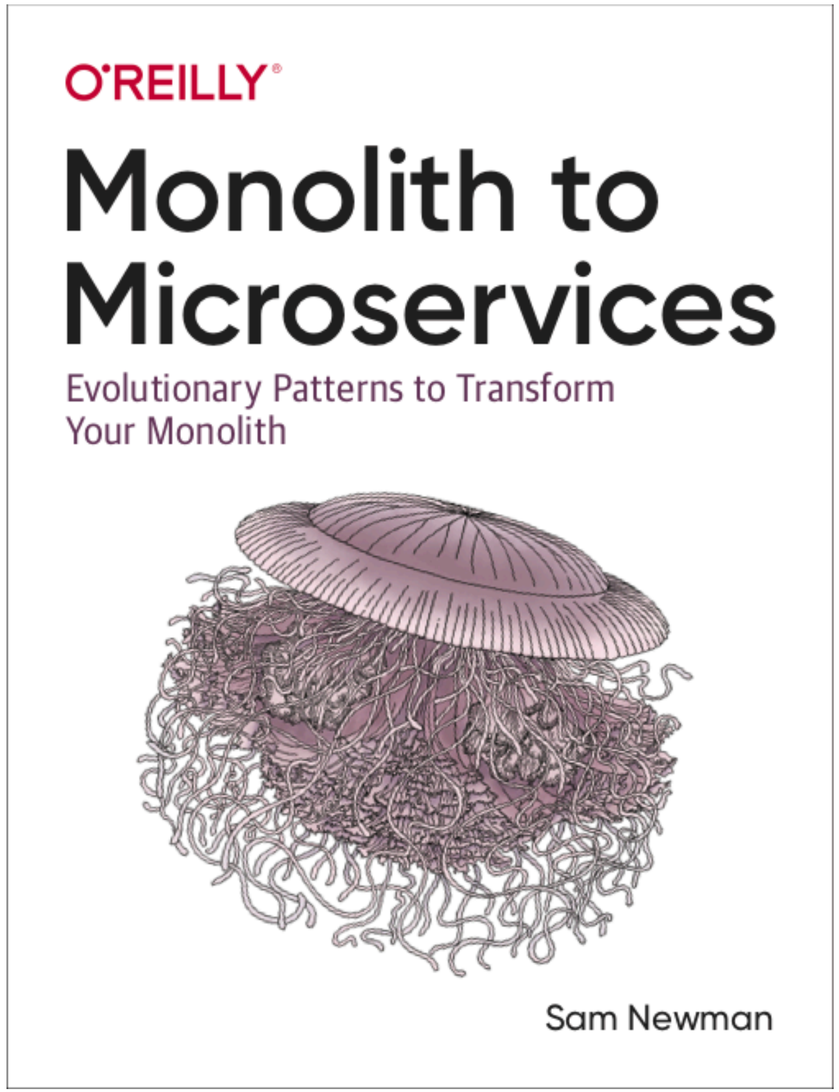
Sam Newman

https://www.flickr.com/photos/snowpeak/42068450985/

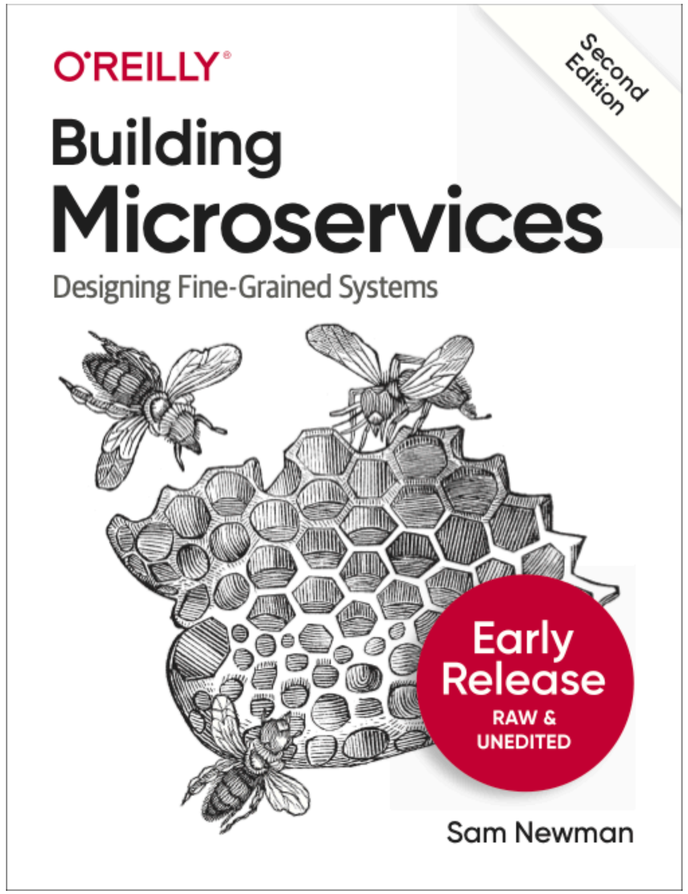
Sam
Newman
& Associates

BOOKS


O'REILLY
Building Microservices
DESIGNING FINE-GRAINED SYSTEMS
Sam Newman


O'REILLY
Monolith to Microservices
Evolutionary Patterns to Transform Your Monolith
Sam Newman

2ND EDITION OF BUILDING MICROSERVICES



Building Microservices, 2nd Edition.

Building Microservices, 2nd Edition, is a forthcoming book on all aspects of Microservices

This book is aimed at giving a broad overview of all aspects related to microservices. Giving a firm foundation in the basics, it goes further, covering off all aspects of the software delivery process, and how they are impacted by adopting a microservice architecture.

What's New?.

This 2nd edition involves every page of the best selling 1st edition being reviewed and where appropriate reworked, with new topics covered for the first time. This edition will be reflecting the shifts in the use of microservices over the last five years, including an exploration of some of the new technology which has emerged.

Status.

Currently, the book is in a very early form, with some chapters available for review at O'Reilly online. These chapters are in a rough form, and are made available to get feedback to help improve the final book.

New chapters will be released in rough form as they become available.

Read on O'Reilly Learning Online

https://samnewman.io/books/building_microservices_2nd_edition/

Three sessions over three weeks

Week 1 - Fundamentals

OVERVIEW

Week 1 - Fundamentals

Week 2 - Adoption and Application Decomposition

OVERVIEW

Week 1 - Fundamentals

Week 2 - Adoption and Application Decomposition

Week 3 - Further Decomposition and …?

Each session is three hours long

Each session is three hours long

We will have two 10-15min breaks

Please ask questions using the Q&A widget

What are microservices?

WHAT WE'LL COVER TODAY

What are microservices?

Independent Deployability

WHAT WE'LL COVER TODAY

What are microservices?

Independent Deployability

Reasons to use them

WHAT WE'LL COVER TODAY

What are microservices?

Independent Deployability

Reasons to use them

Reasons not to!

POLL: WHAT'S YOUR EXPERIENCE WITH MICROSERVICES?

Never used them!

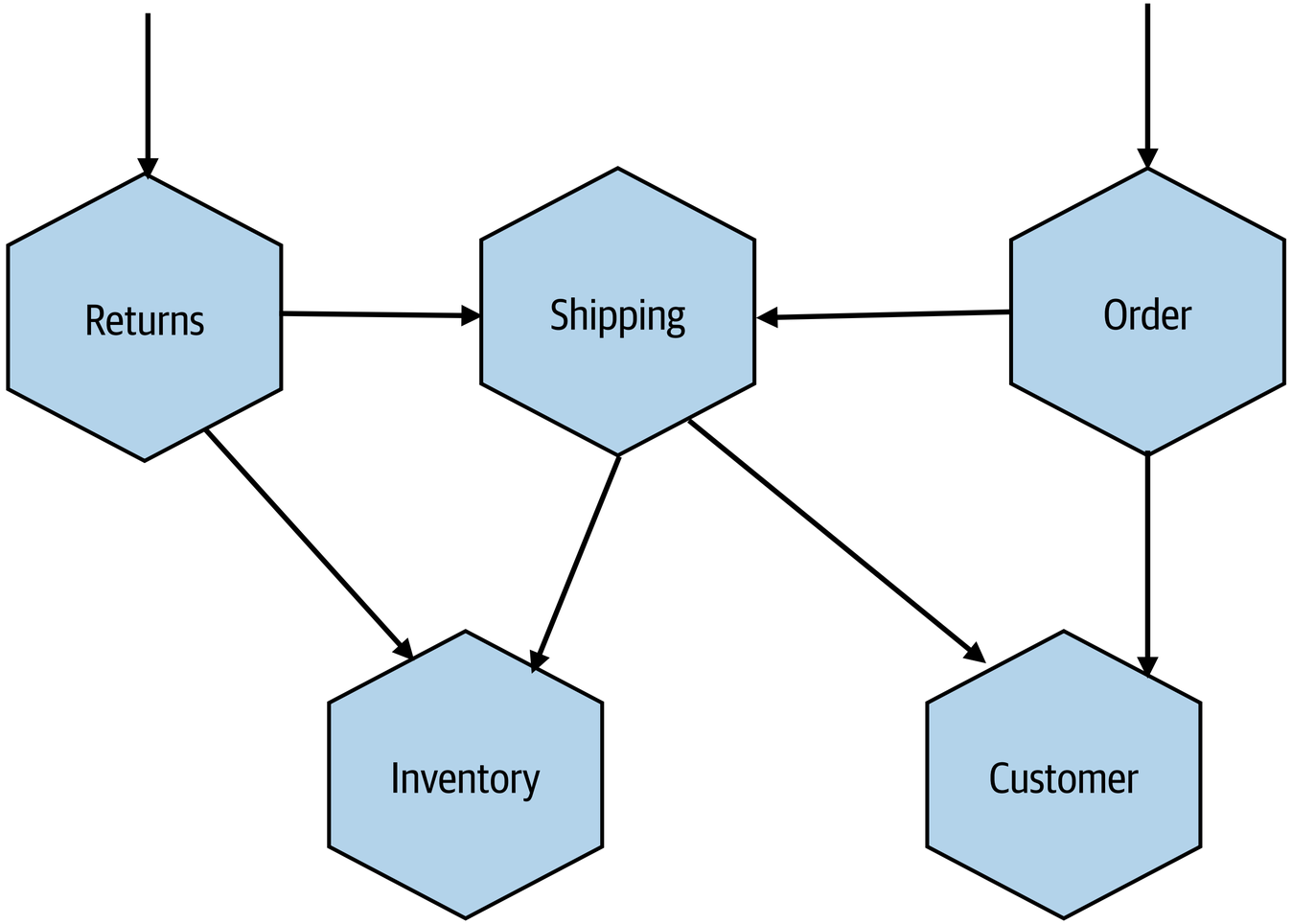
Just started (less than 1 year experience)

Using them for a bit (1-3 years)

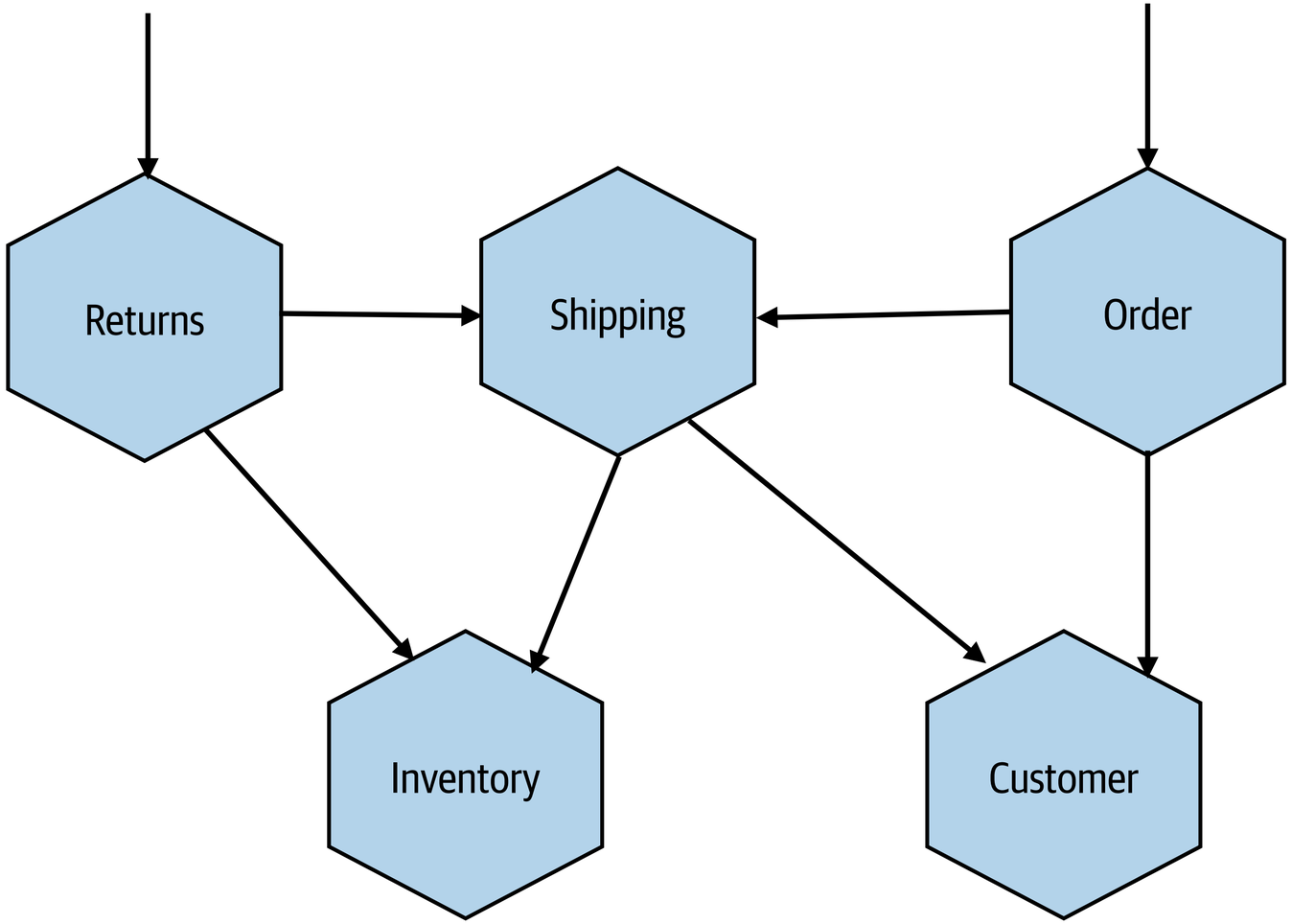
For ages! (3 years+)

1/4 What are microservices?

AT A HIGH LEVEL

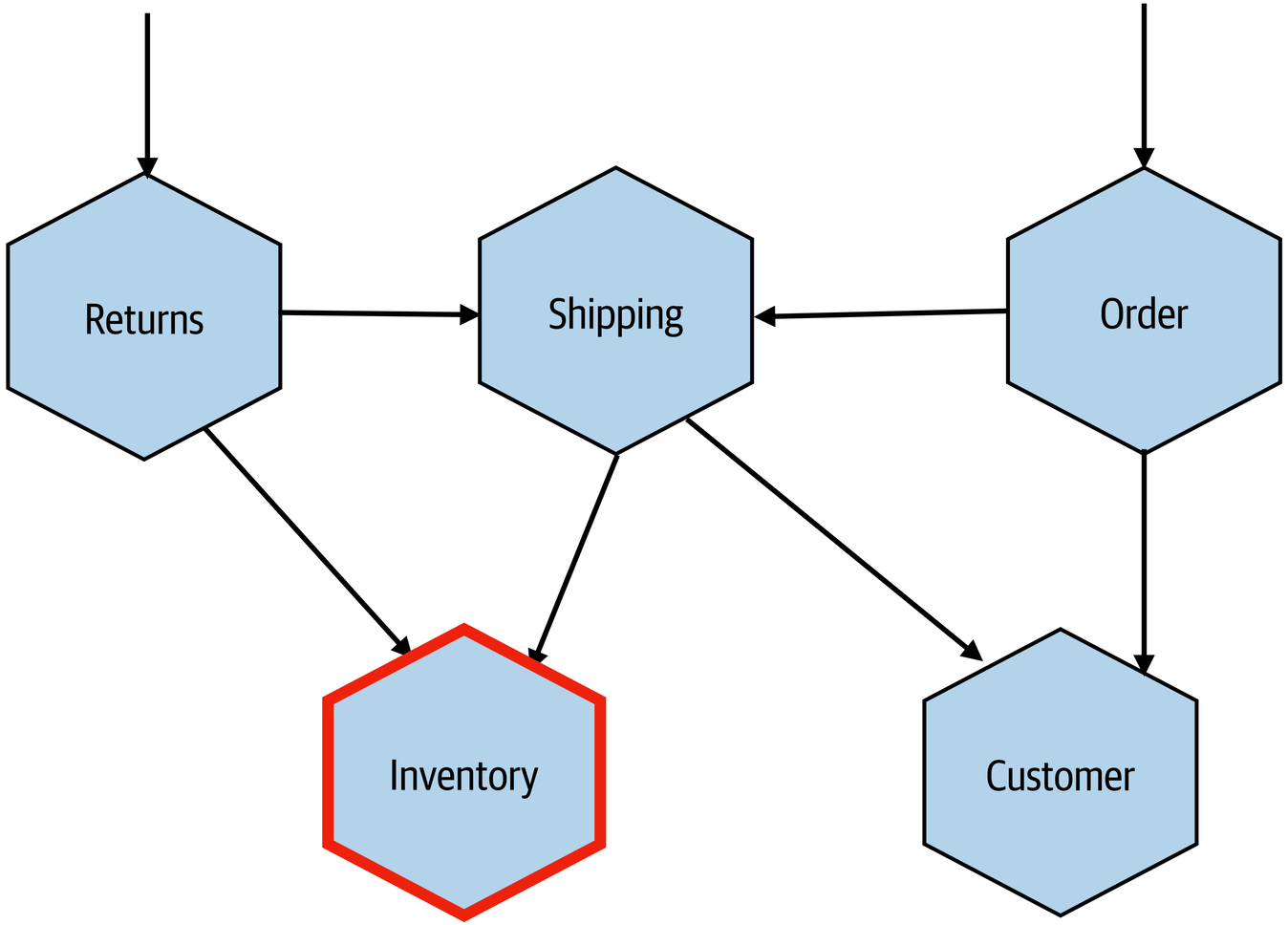


AT A HIGH LEVEL



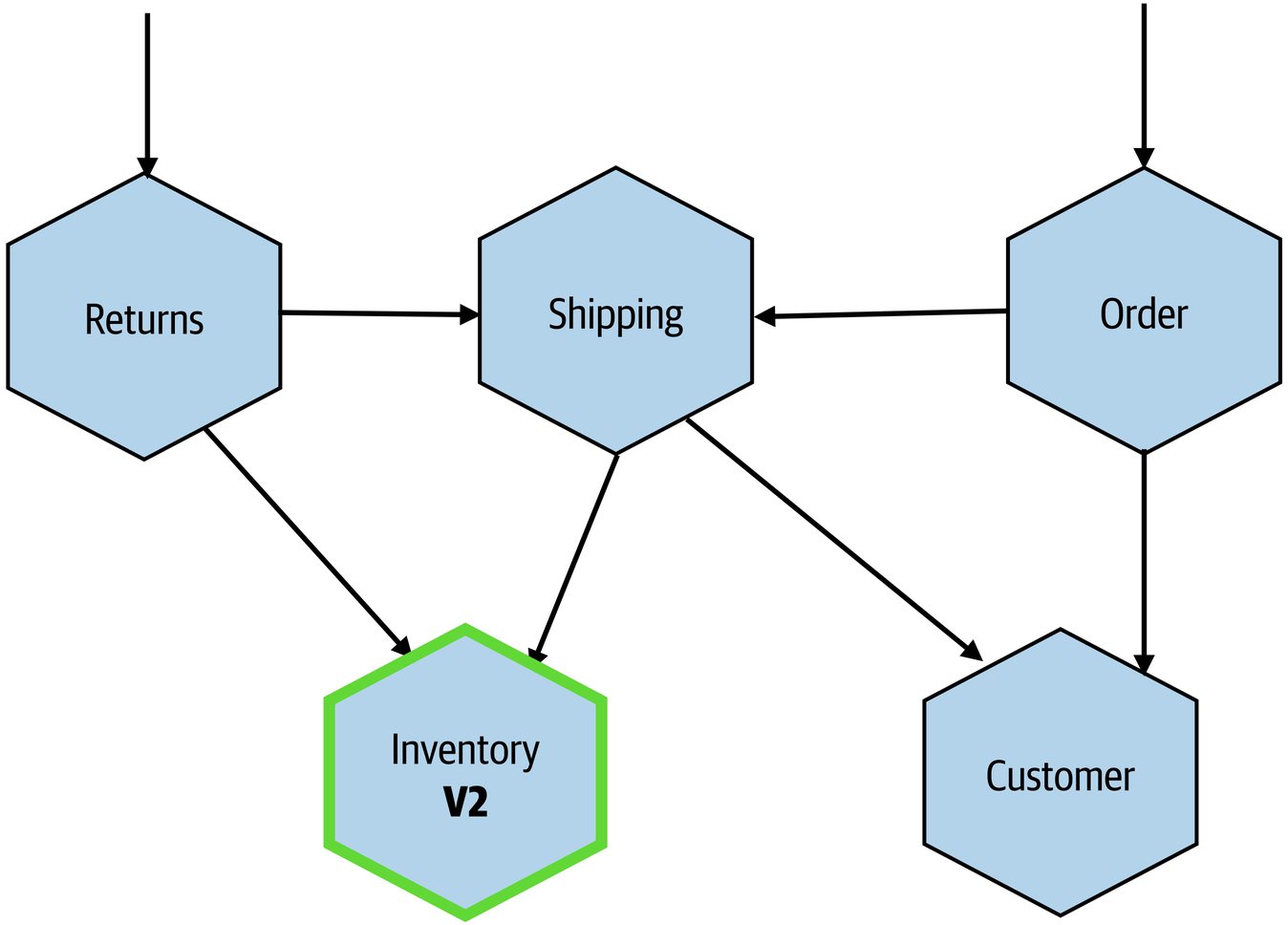
Independently Deployable

AT A HIGH LEVEL



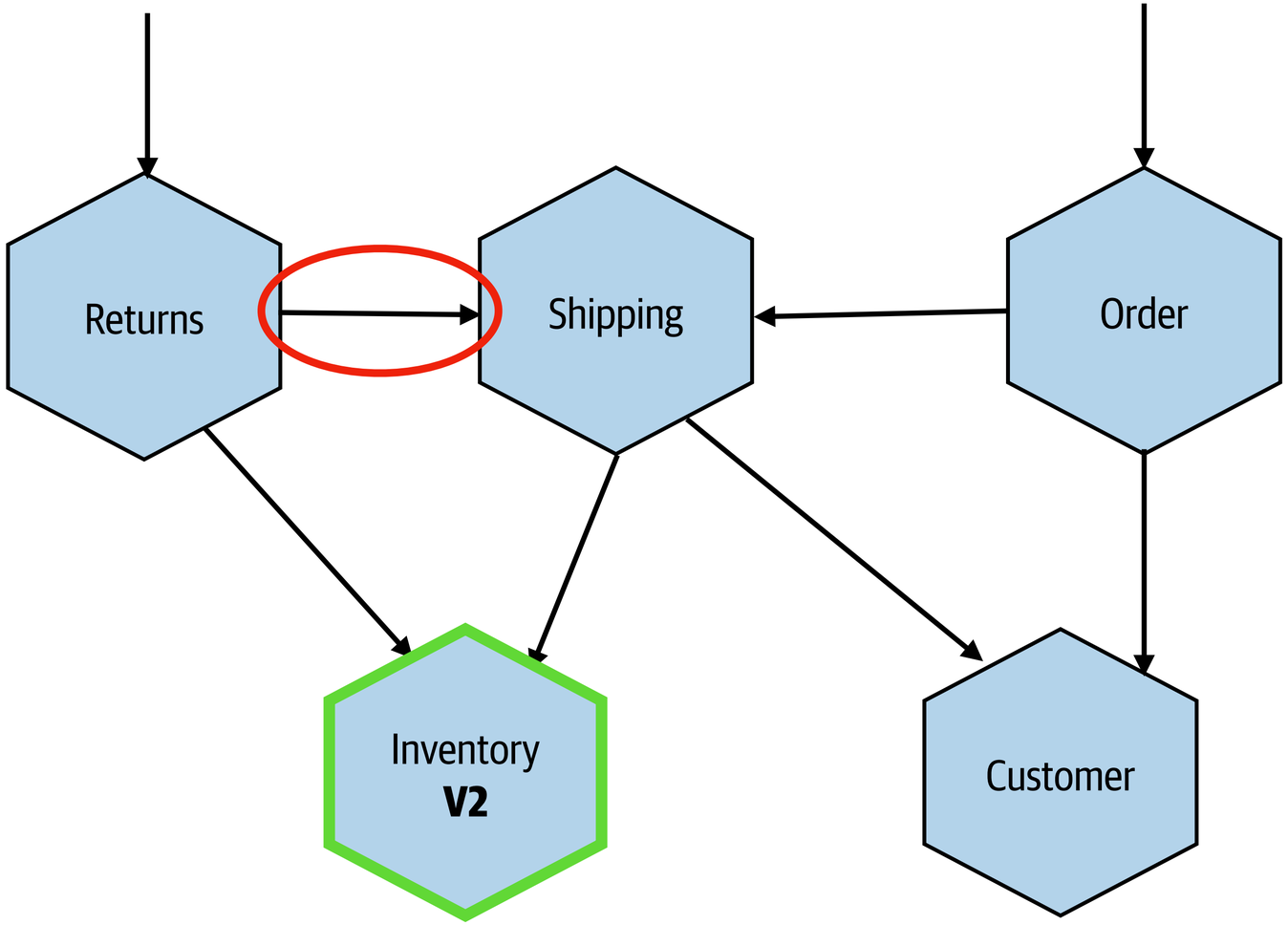
Independently Deployable

AT A HIGH LEVEL



Independently Deployable

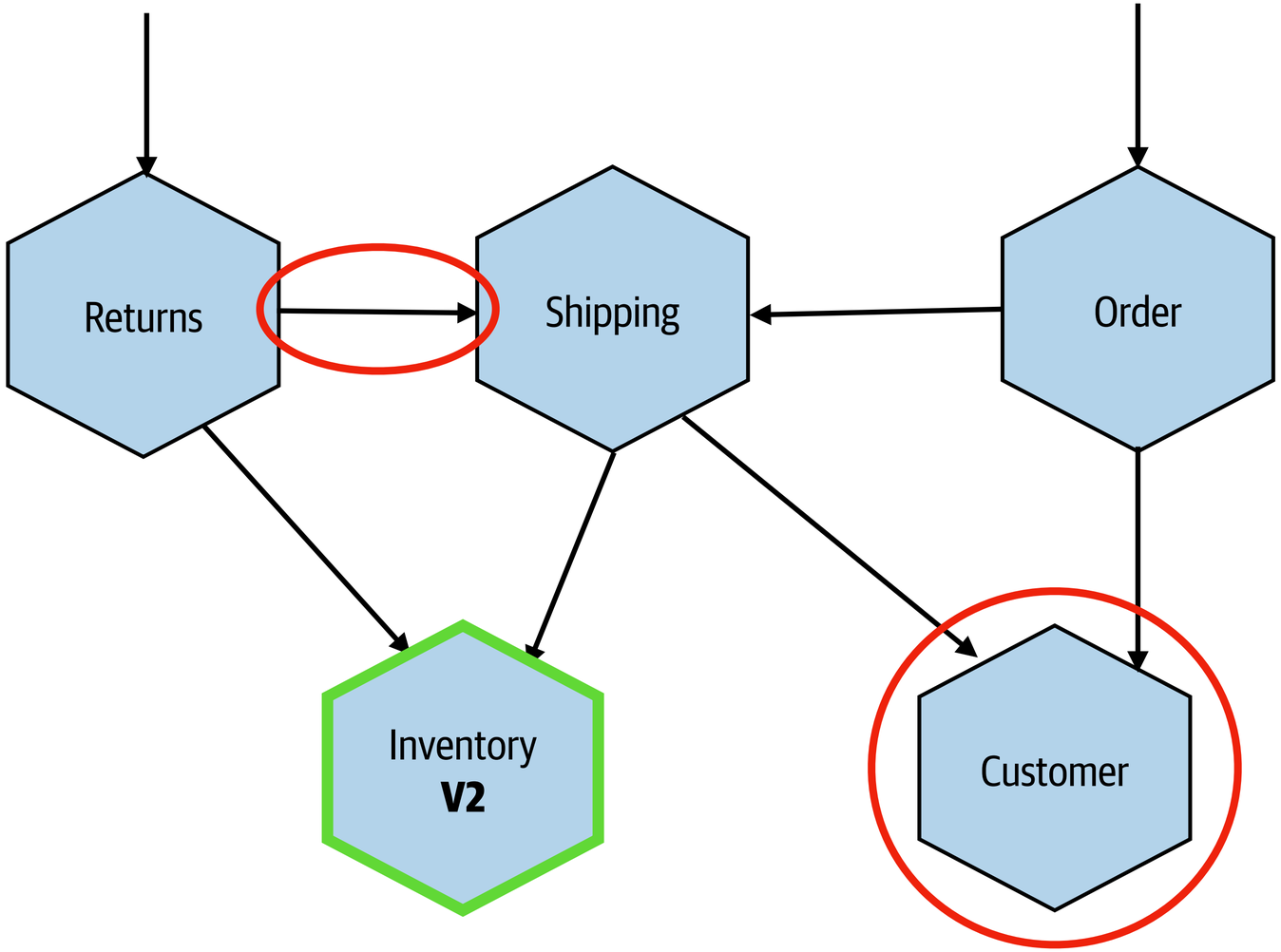
AT A HIGH LEVEL



Independently Deployable

Microservices often depend on other microservices

AT A HIGH LEVEL

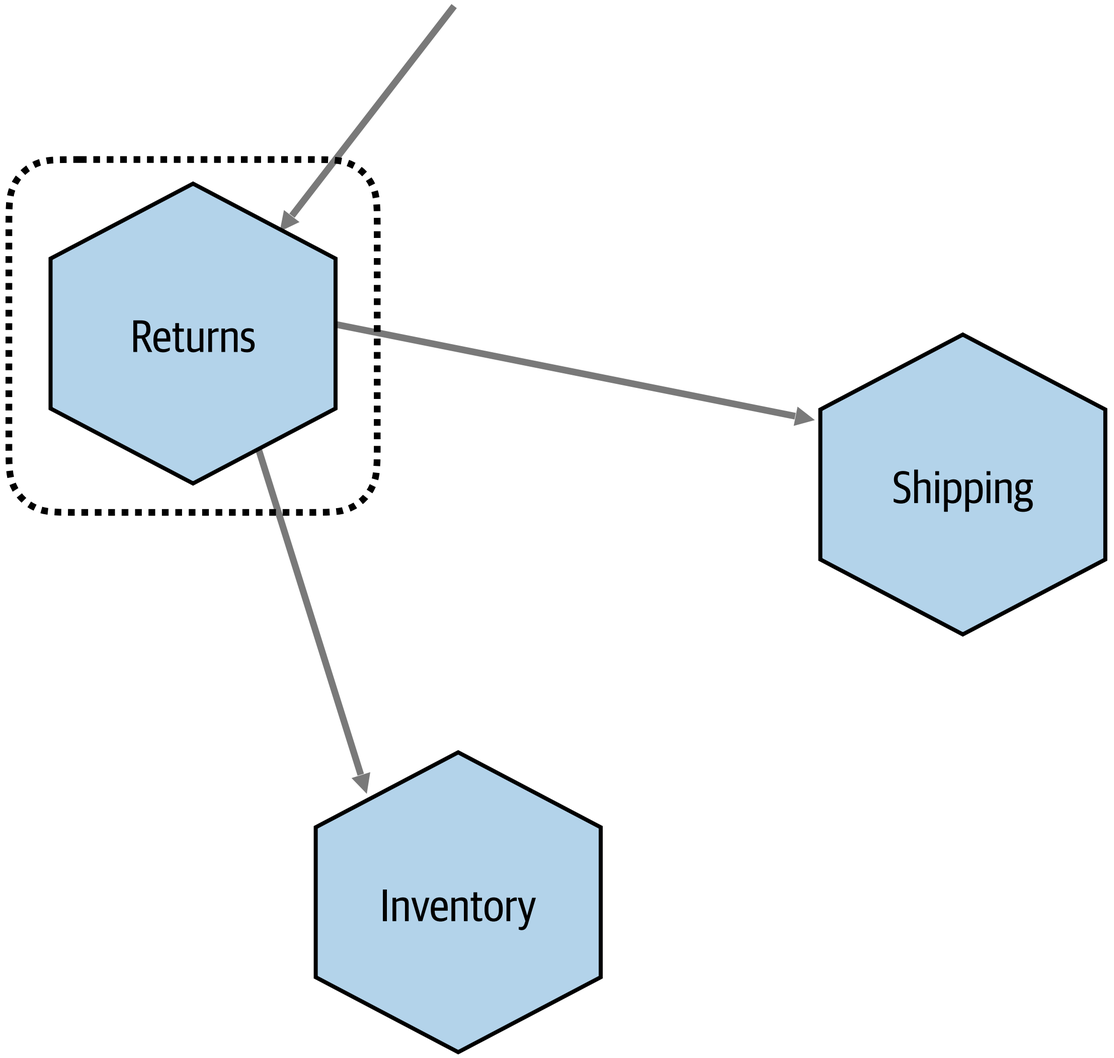


Independently Deployable

Microservices often depend on other microservices

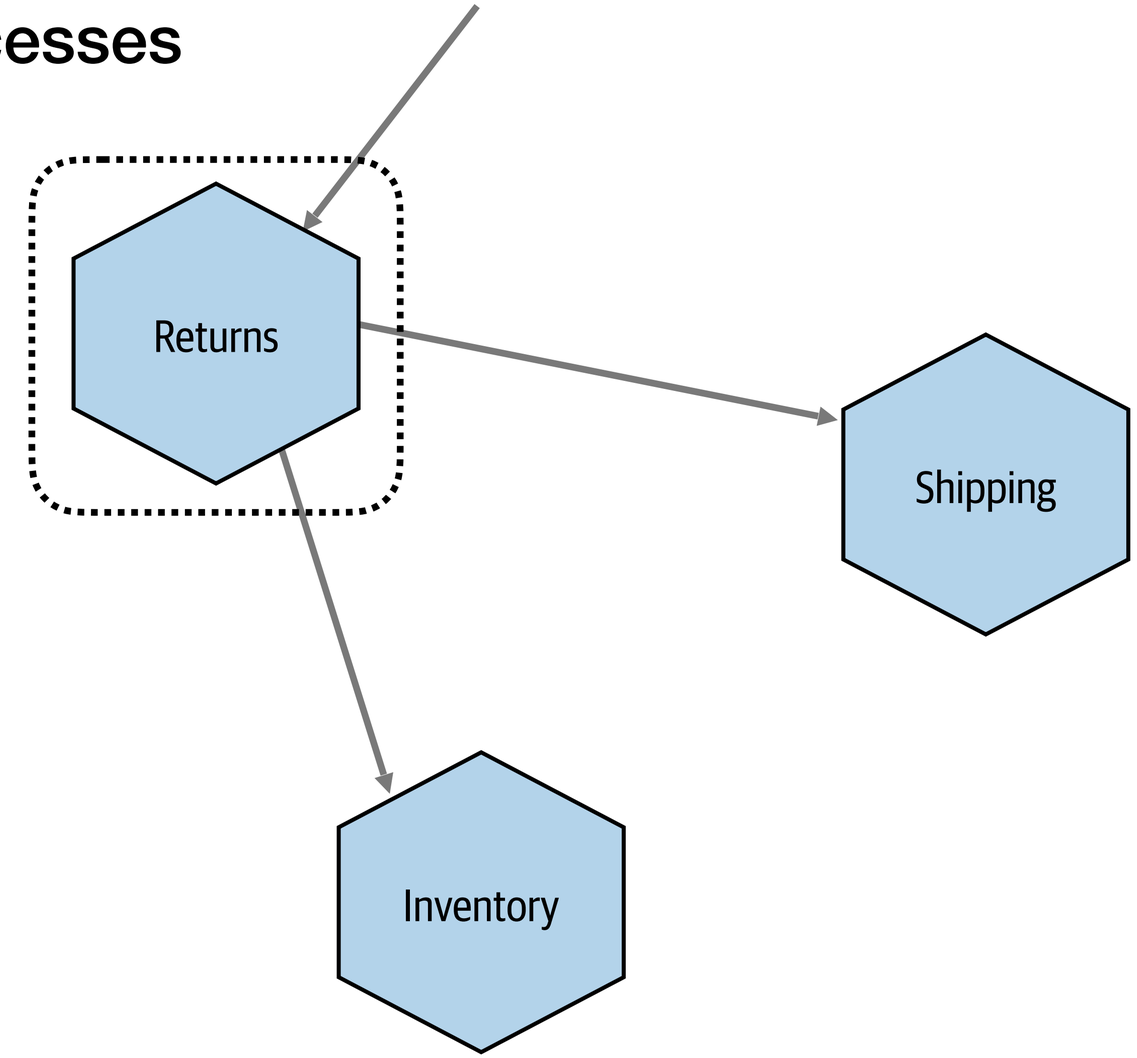
Boundaries primarily defined by the domain

IN ISOLATION



IN ISOLATION

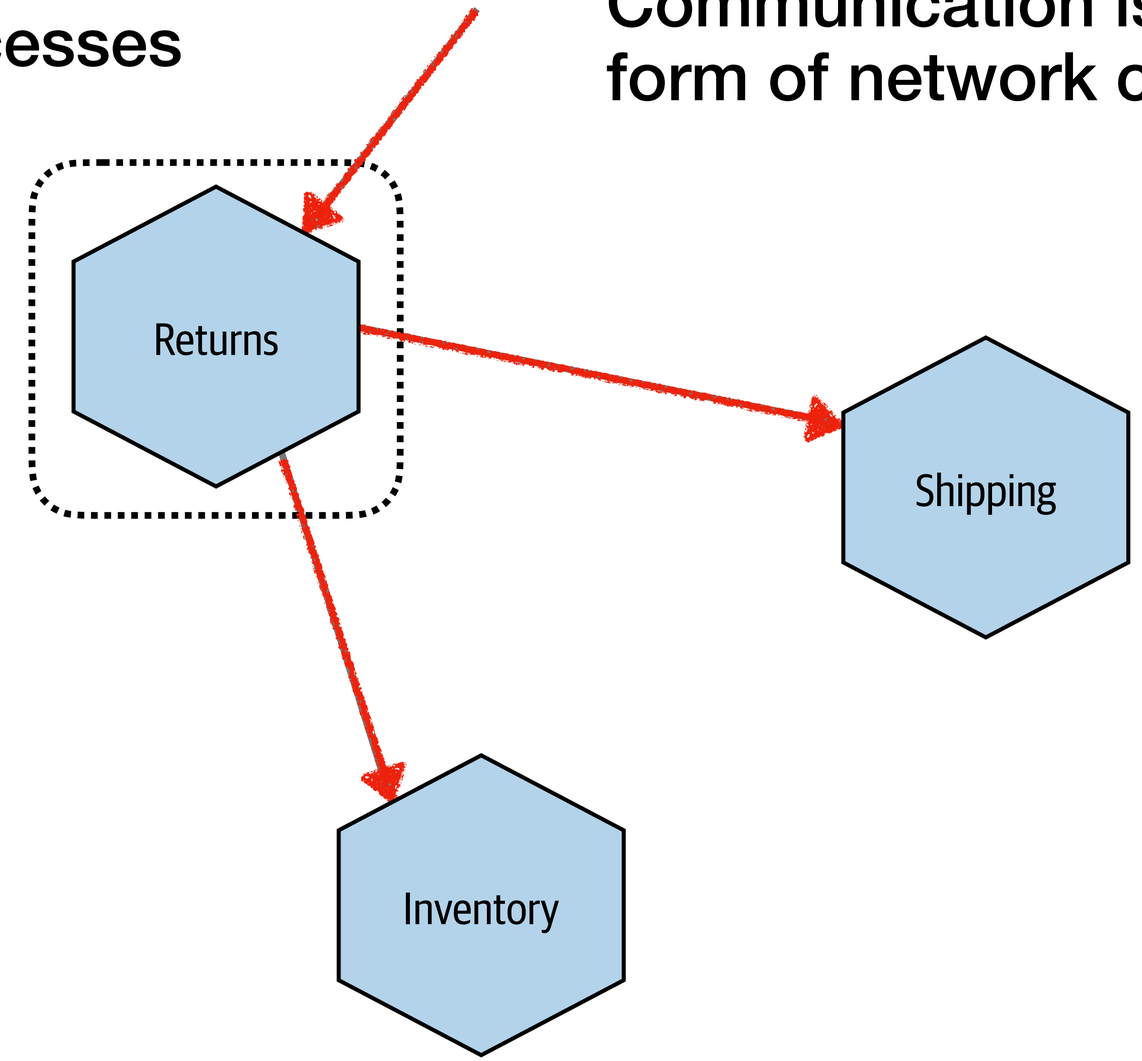
Microservice instances are typically separate processes



IN ISOLATION

Microservice instances are typically separate processes

Communication is via some form of network call

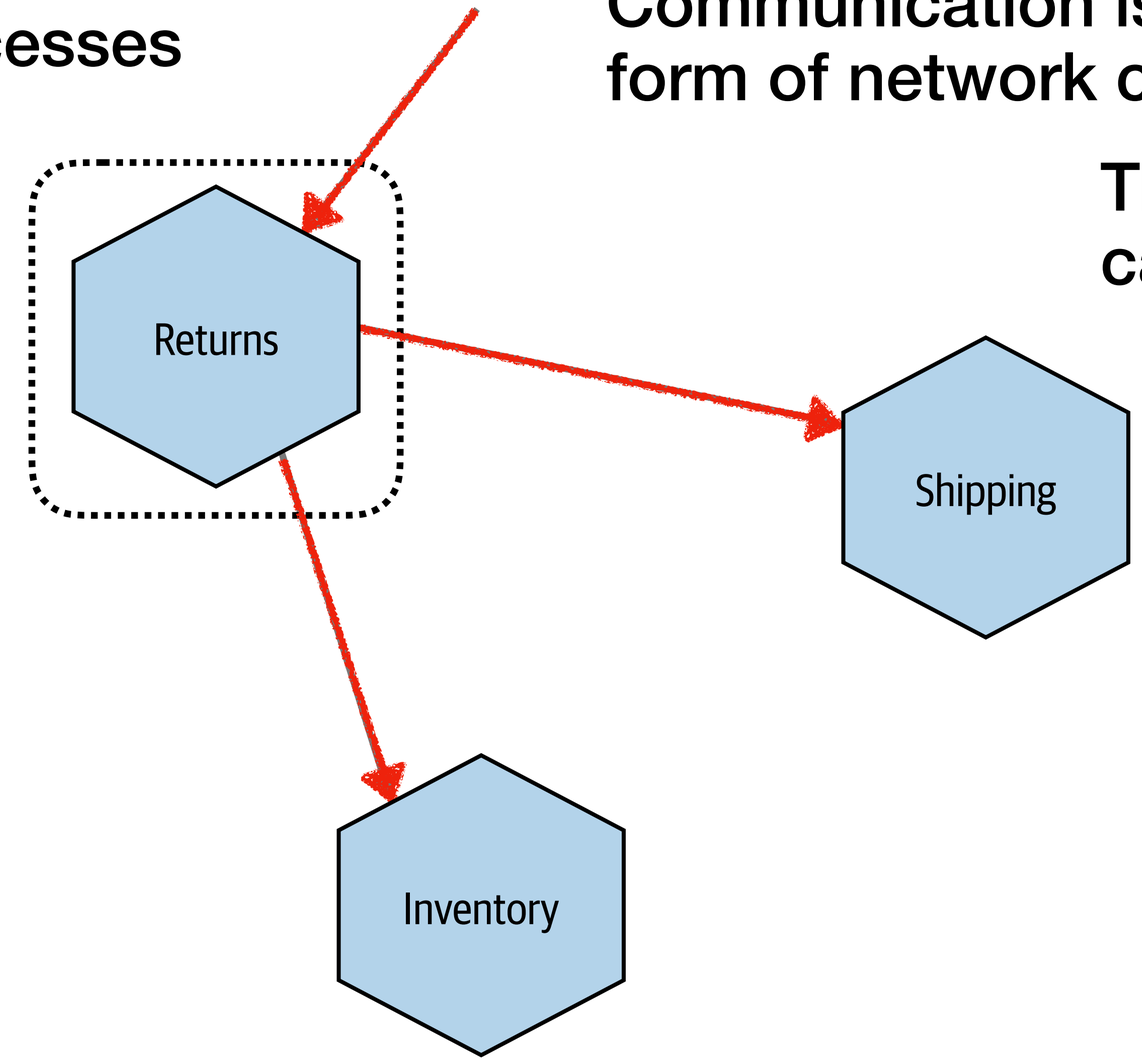


IN ISOLATION

Microservice instances are typically separate processes

Communication is via some form of network call

The exact protocols can vary

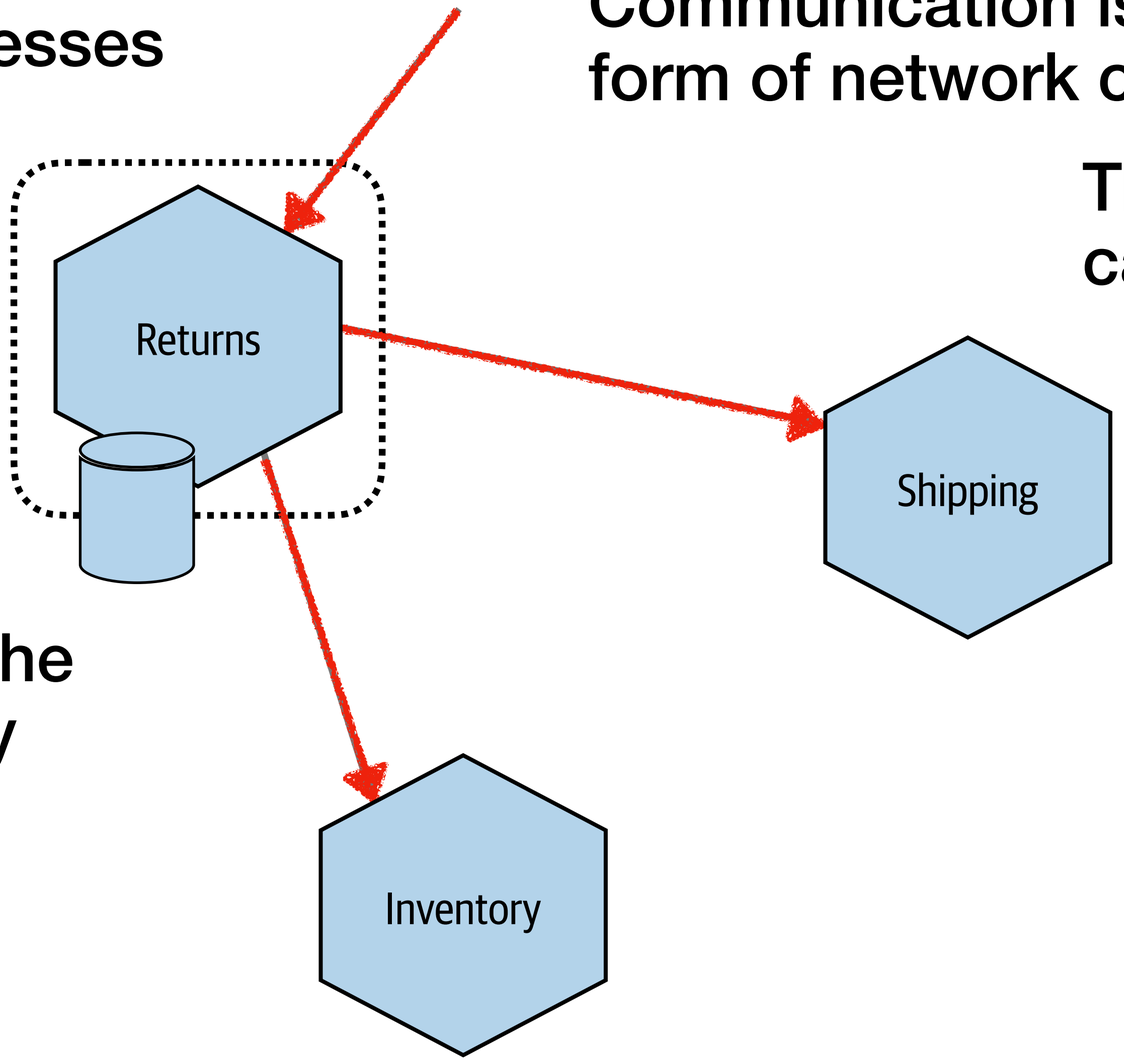


IN ISOLATION

Microservice instances are typically separate processes

Communication is via some form of network call

The exact protocols can vary



Data is hidden inside the microservice boundary

STYLES OF COMMUNICATION

Request/Response

STYLES OF COMMUNICATION

Request/Response

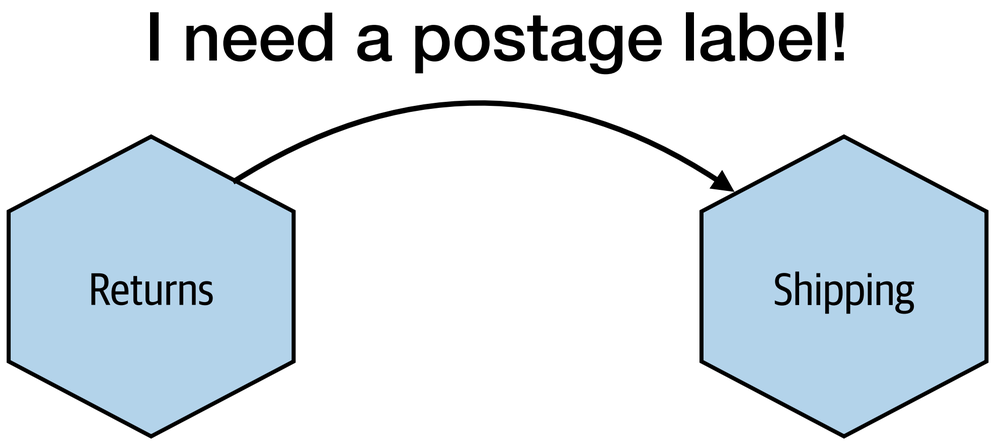
Event-Driven

REQUEST-RESPONSE

The consumer is *asking* the downstream microservice for something

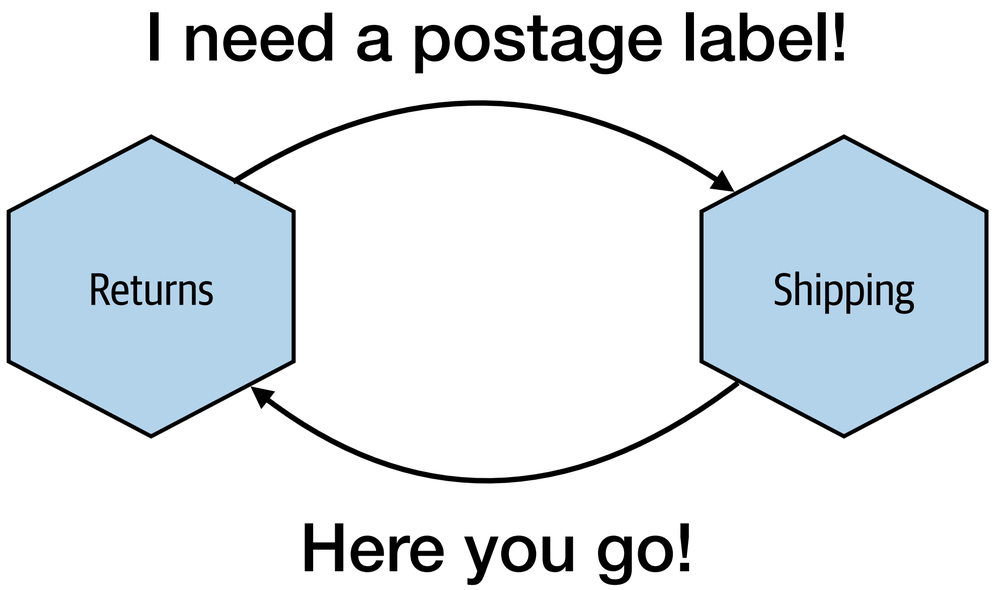
REQUEST-RESPONSE

The consumer is *asking* the downstream microservice for something



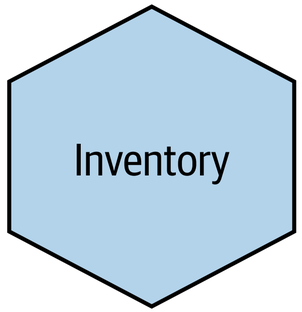
REQUEST-RESPONSE

The consumer is *asking* the downstream microservice for something



EVENT-DRIVEN

An event (fact) is broadcast

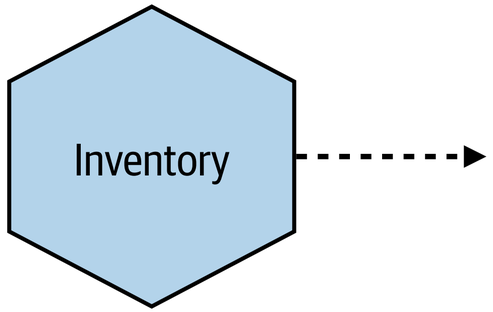


Interested parties receive the event and react accordingly

The emitting microservice doesn’t need to know about consumers

EVENT-DRIVEN

An event (fact) is broadcast

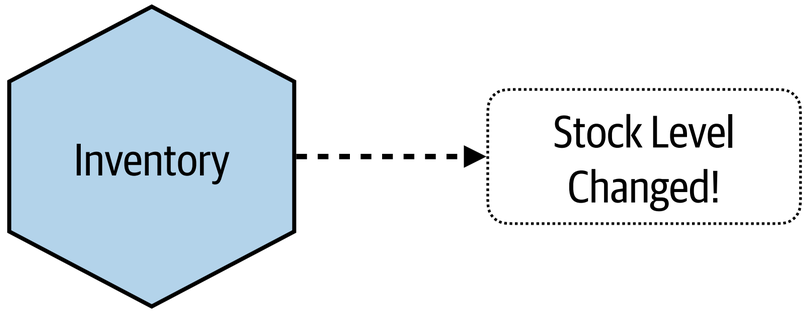


Interested parties receive the event and react accordingly

The emitting microservice doesn’t need to know about consumers

EVENT-DRIVEN

An event (fact) is broadcast

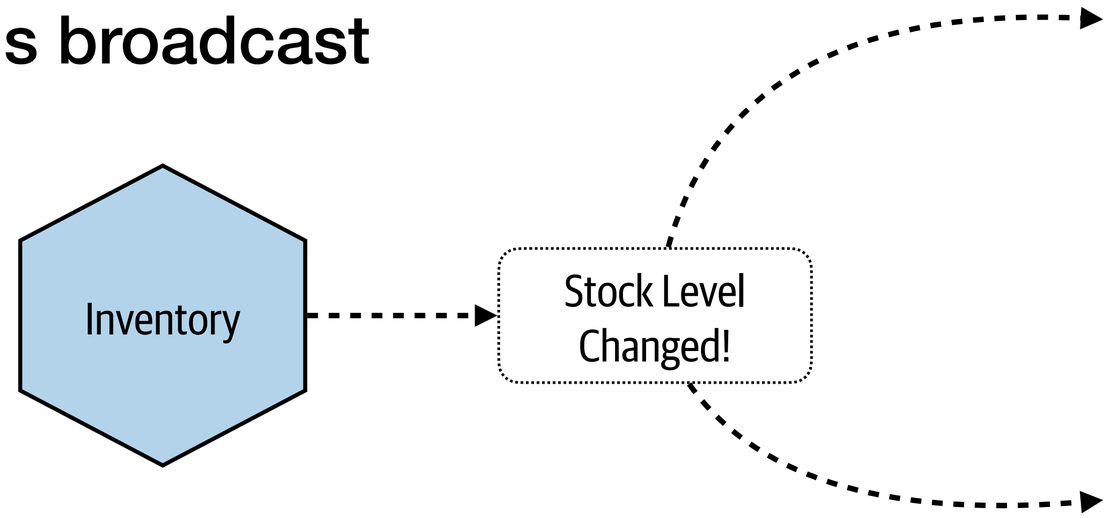


Interested parties receive the event and react accordingly

The emitting microservice doesn’t need to know about consumers

EVENT-DRIVEN

An event (fact) is broadcast

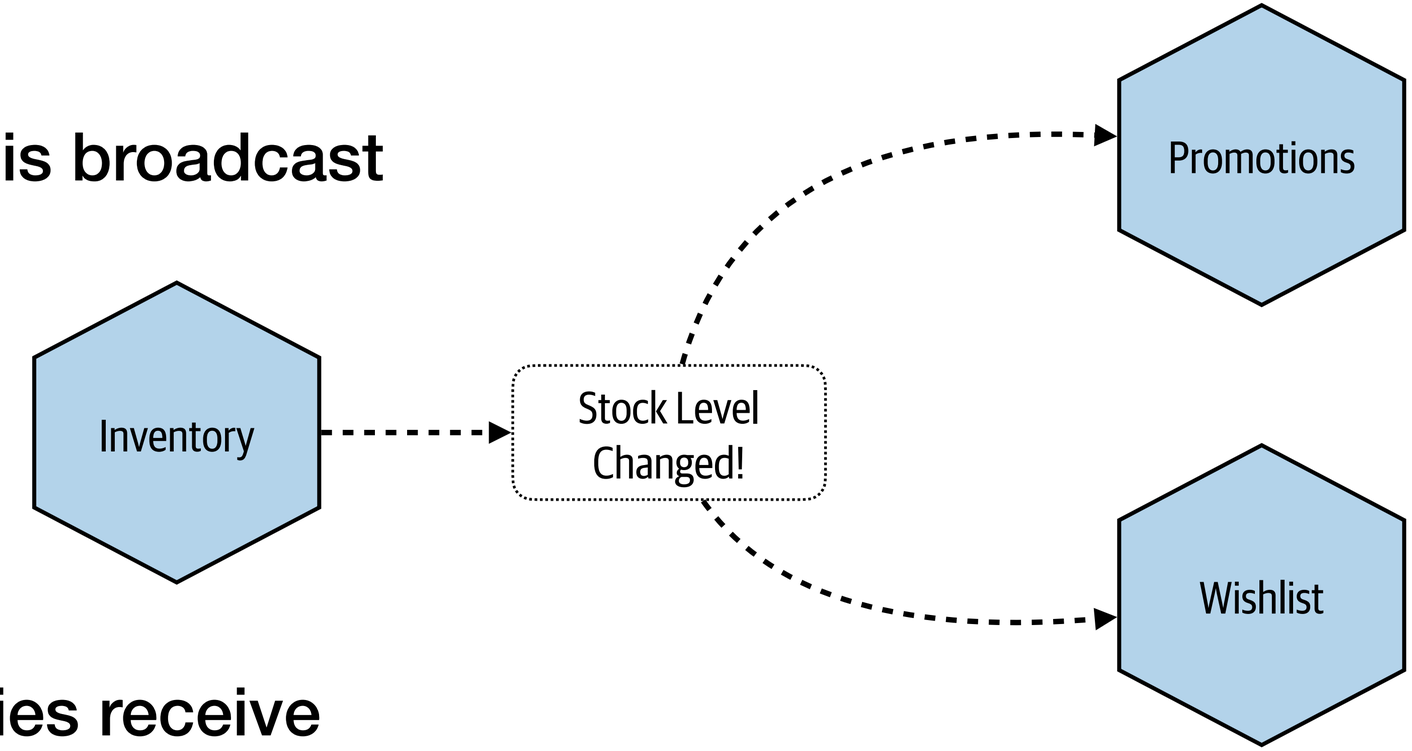


Interested parties receive the event and react accordingly

The emitting microservice doesn't need to know about consumers

EVENT-DRIVEN

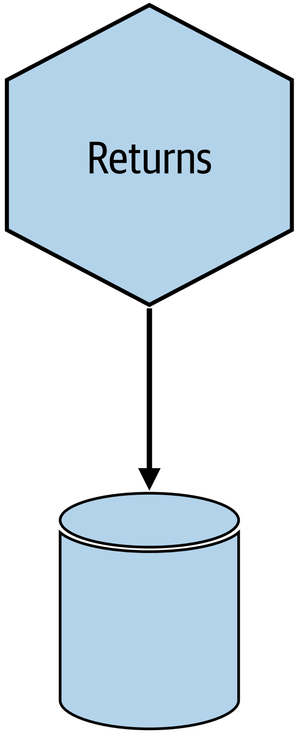
An event (fact) is broadcast



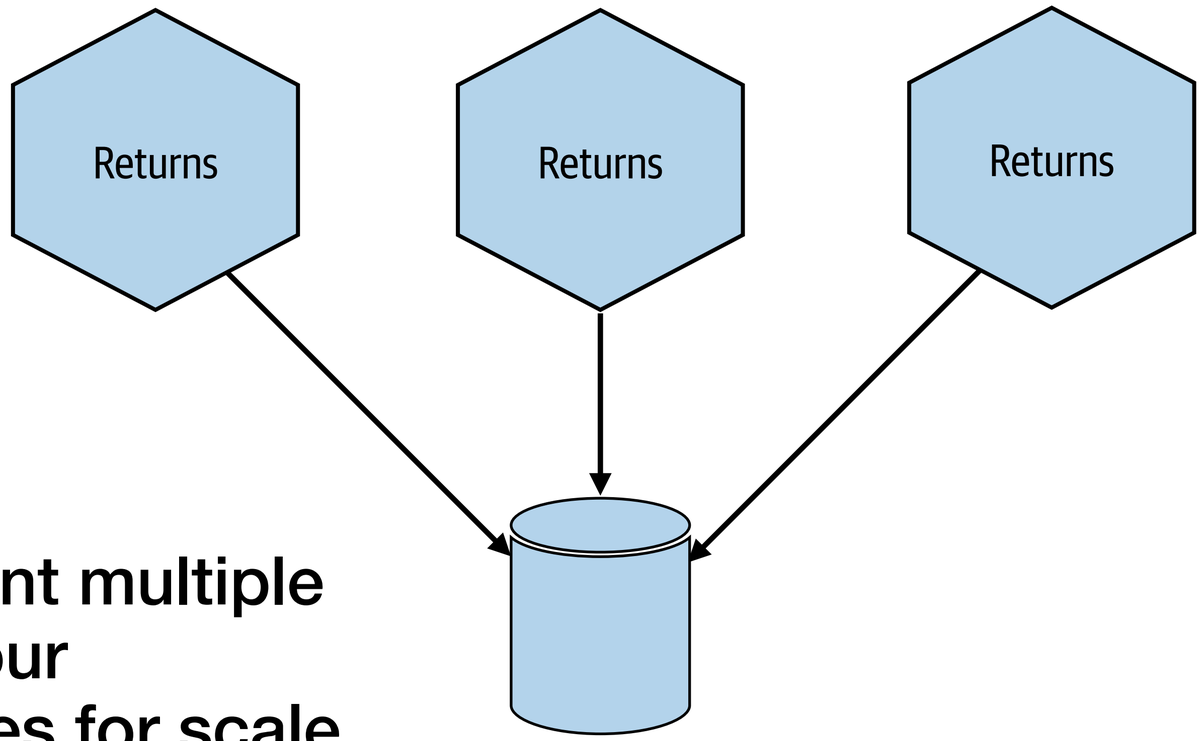
Interested parties receive the event and react accordingly

The emitting microservice doesn't need to know about consumers

FROM LOGICAL TO PHYSICAL



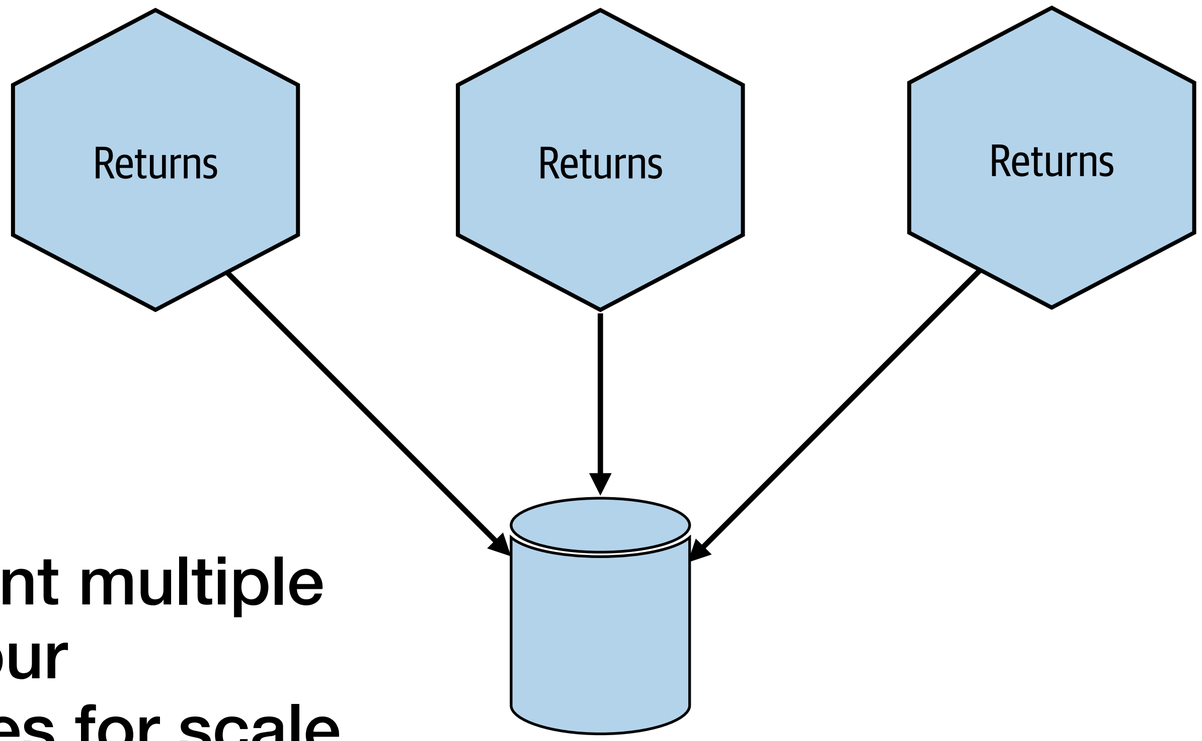
FROM LOGICAL TO PHYSICAL



You may want multiple copies of your microservices for scale or robustness reasons

FROM LOGICAL TO PHYSICAL

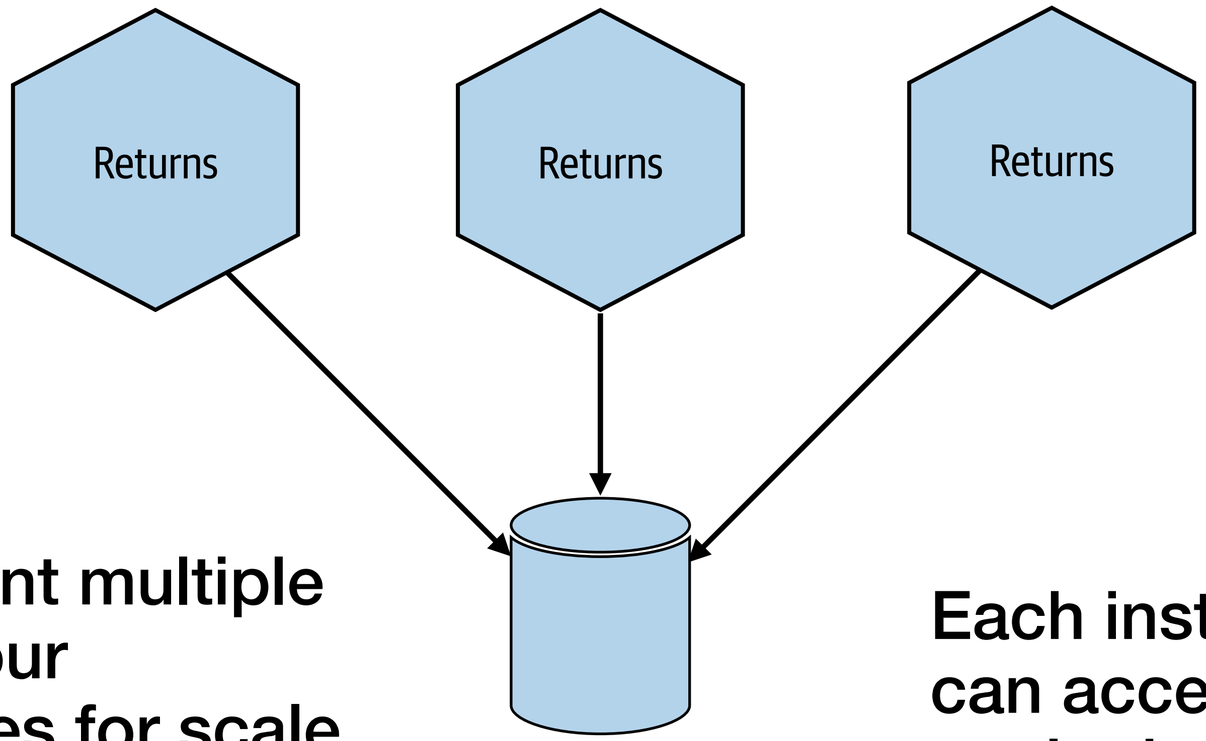
Instances typically deployed on different “machines”



You may want multiple copies of your microservices for scale or robustness reasons

FROM LOGICAL TO PHYSICAL

Instances typically deployed on different “machines”

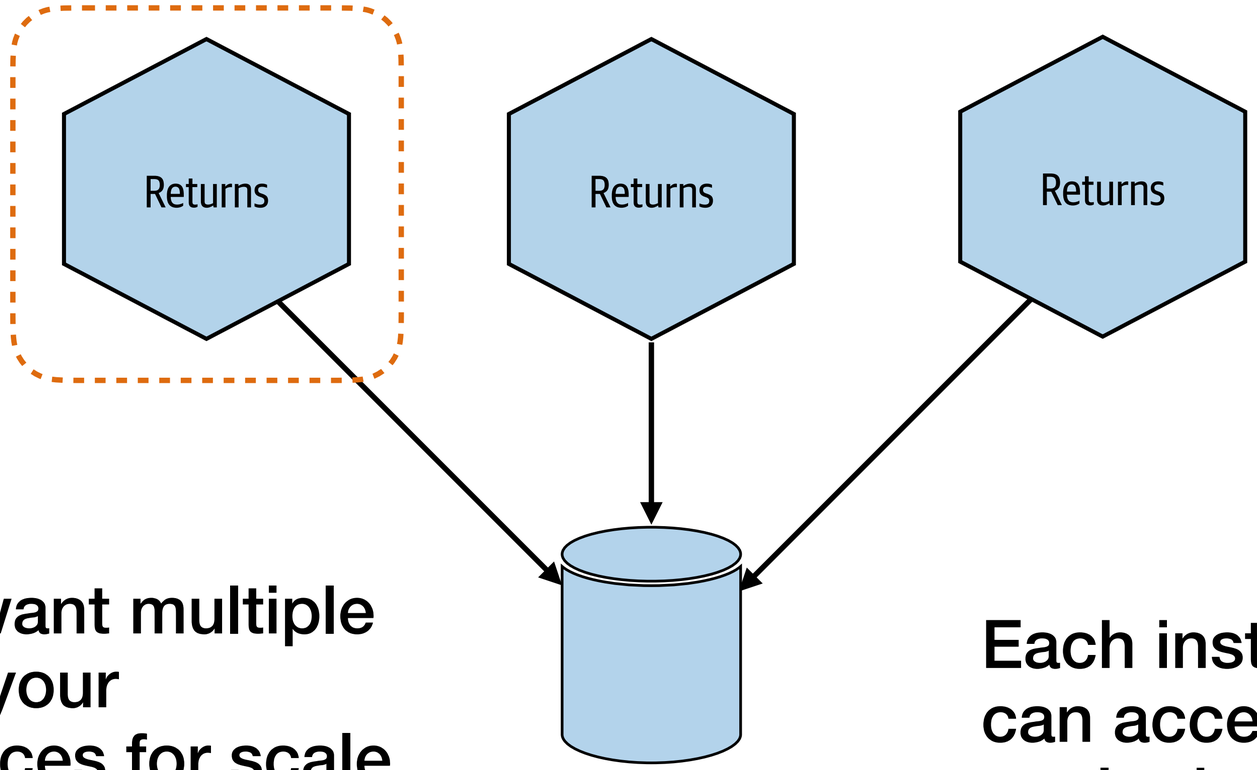


You may want multiple copies of your microservices for scale or robustness reasons

Each instance of a microservice can access the same DB - they are logically all part of the same thing

FROM LOGICAL TO PHYSICAL

Instances typically deployed on different “machines”

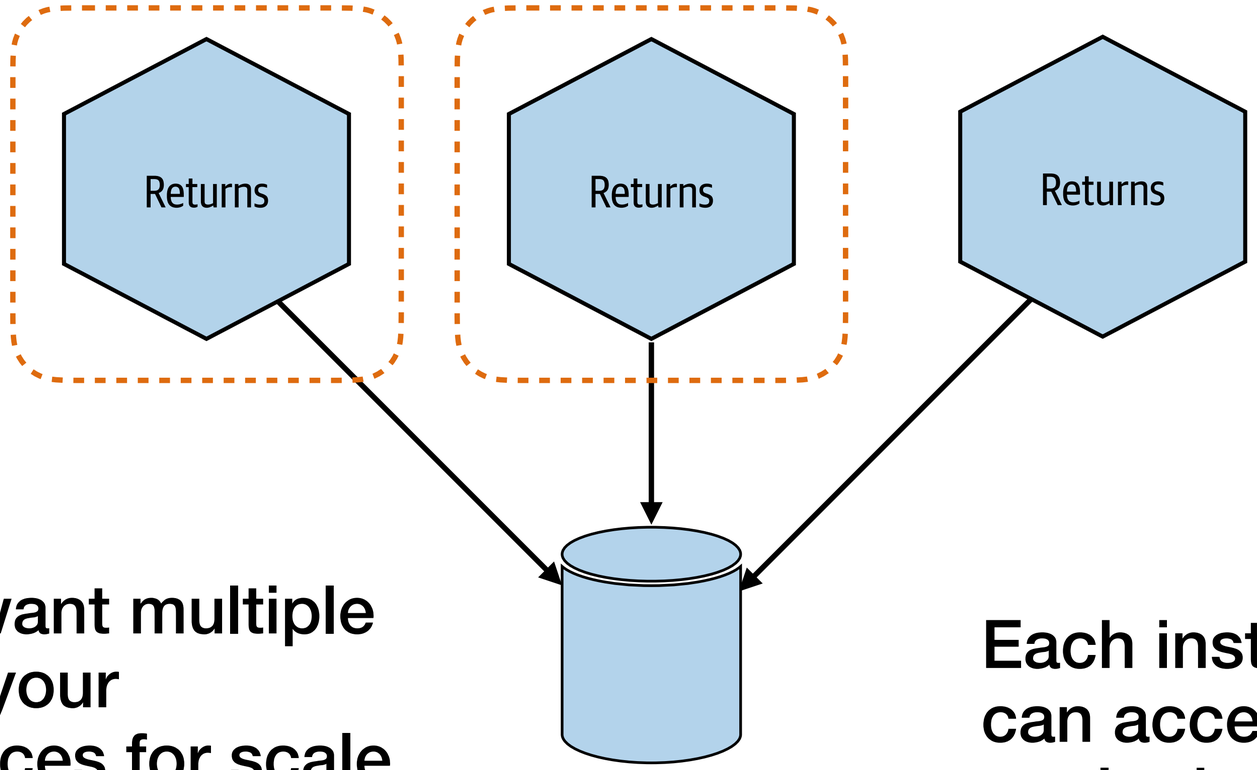


You may want multiple copies of your microservices for scale or robustness reasons

Each instance of a microservice can access the same DB - they are logically all part of the same thing

FROM LOGICAL TO PHYSICAL

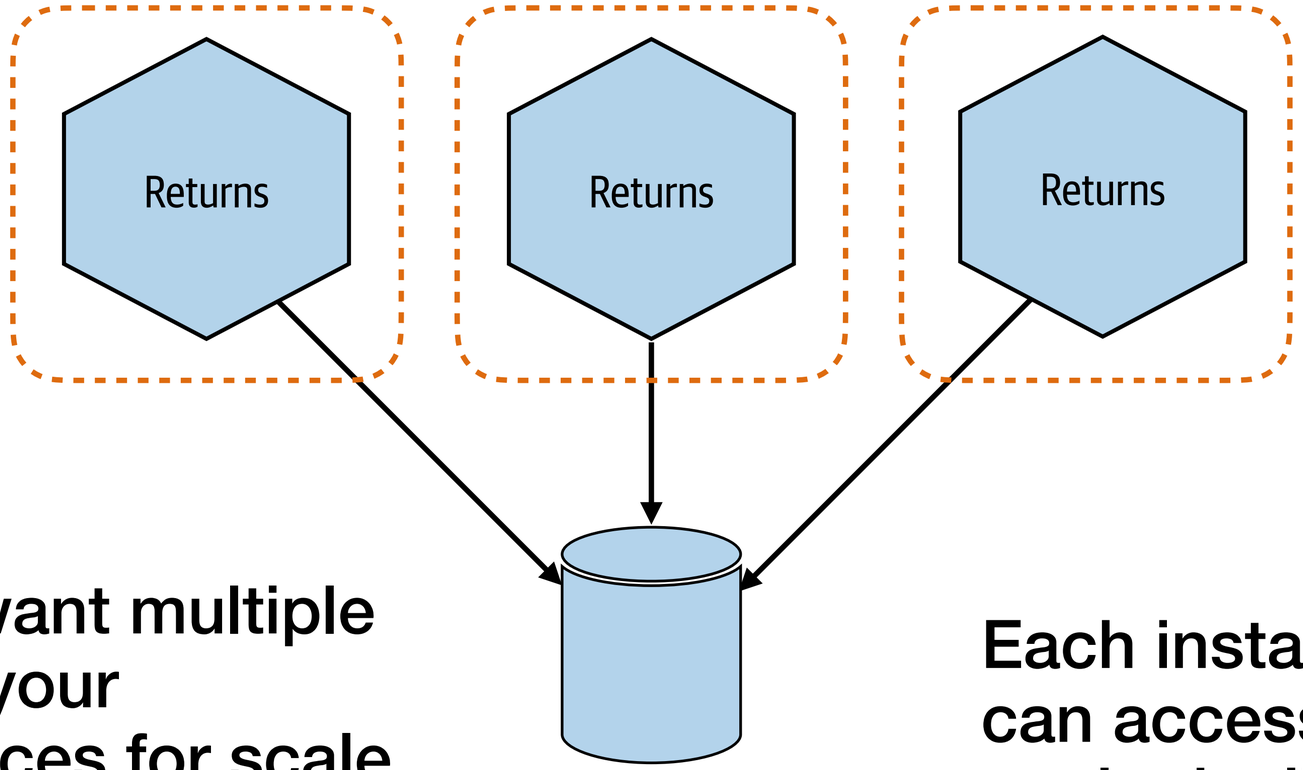
Instances typically deployed on different “machines”



You may want multiple copies of your microservices for scale or robustness reasons

Each instance of a microservice can access the same DB - they are logically all part of the same thing

FROM LOGICAL TO PHYSICAL

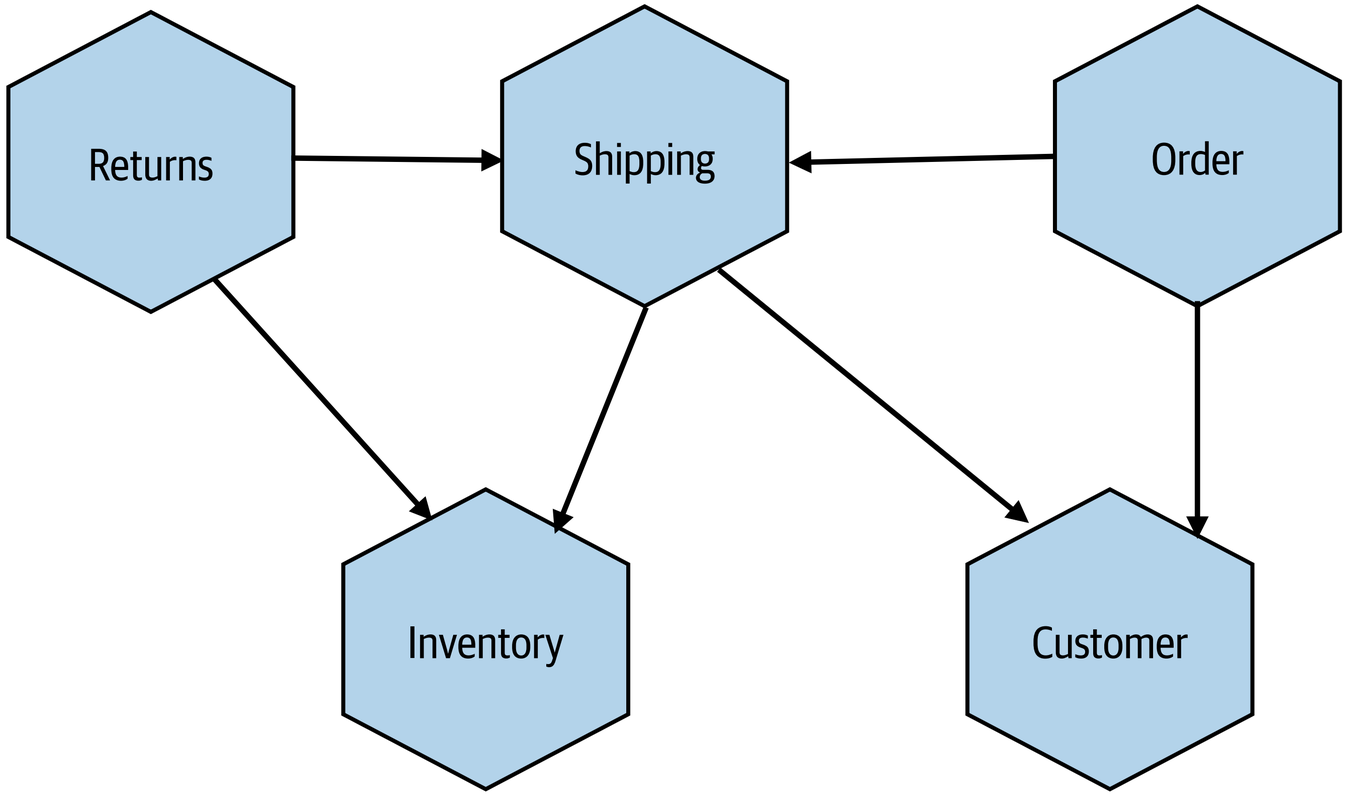


Instances typically deployed on different “machines”

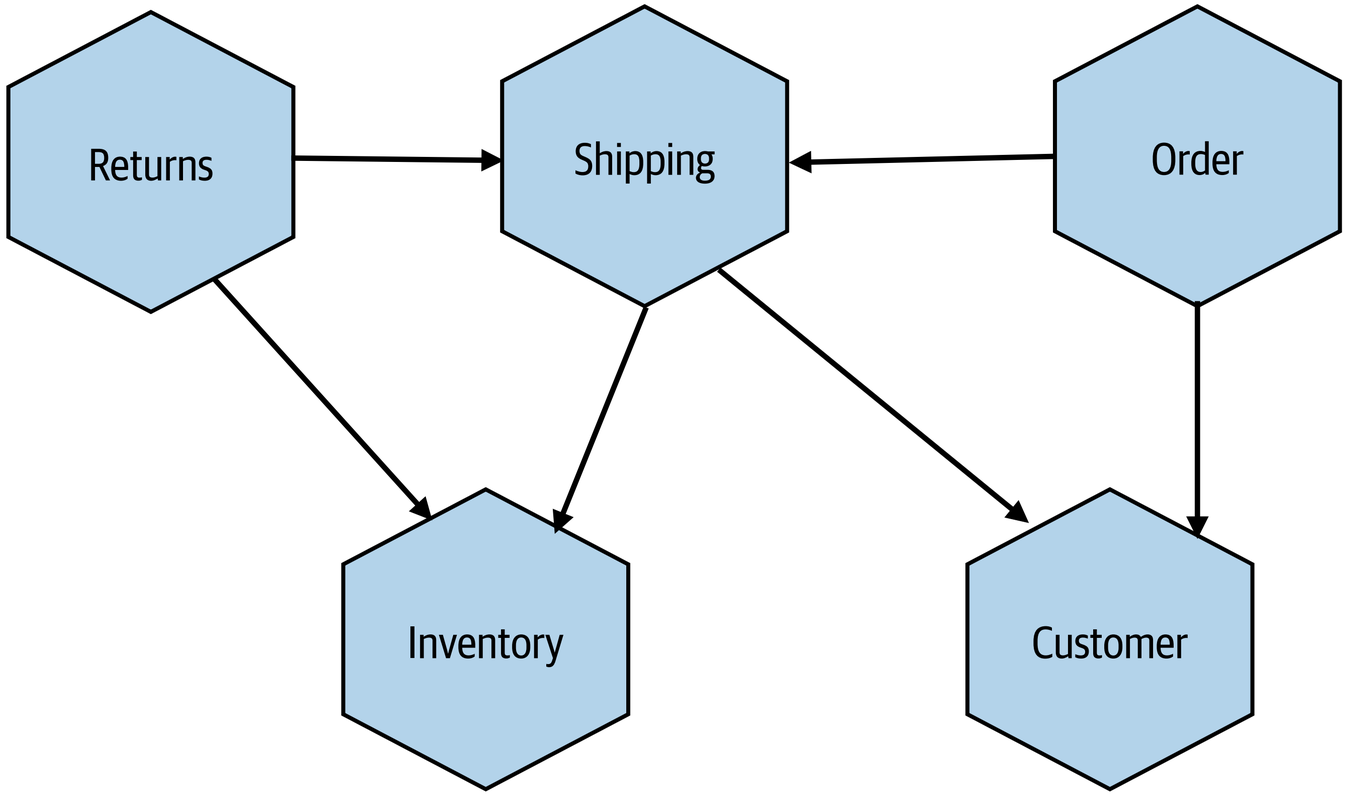
You may want multiple copies of your microservices for scale or robustness reasons

Each instance of a microservice can access the same DB - they are logically all part of the same thing

DATABASES

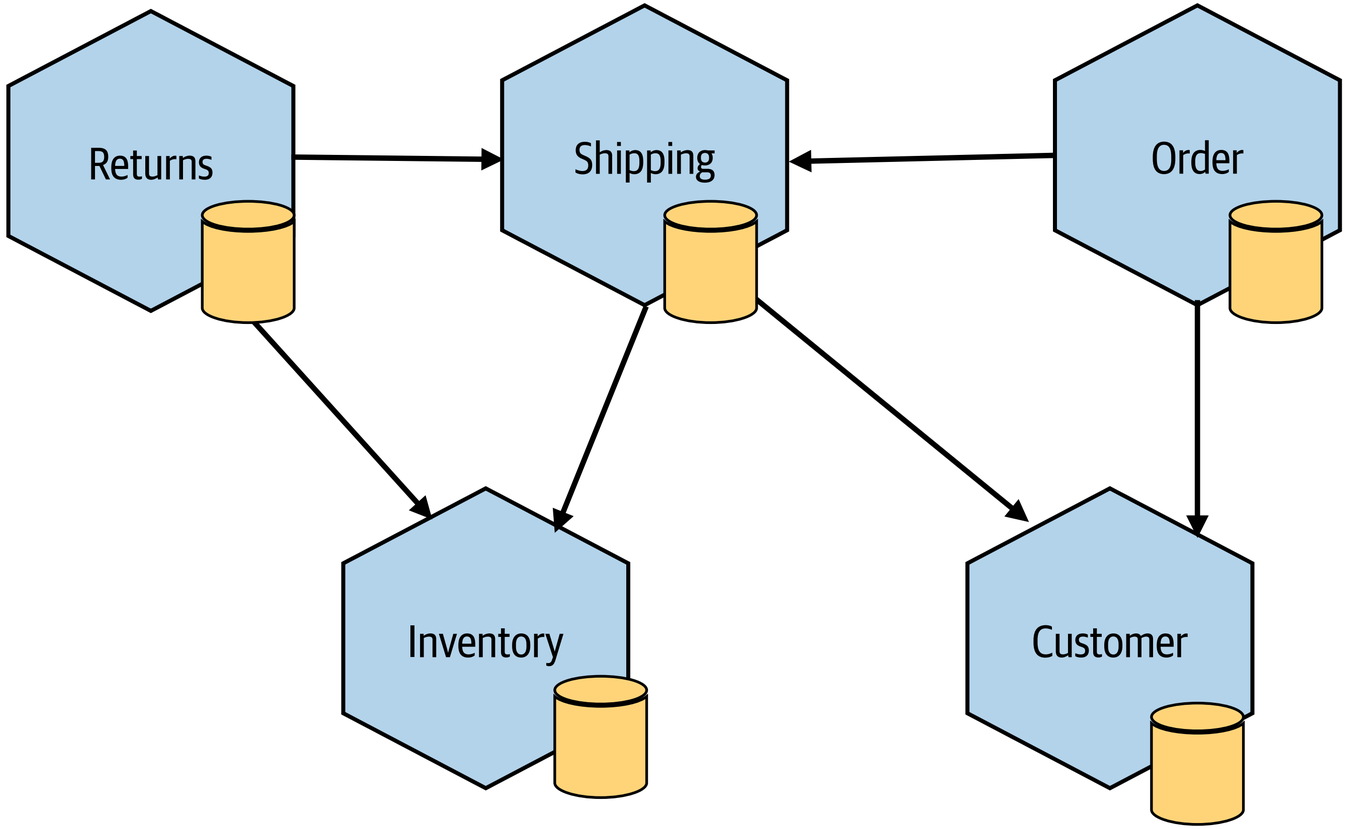


DATABASES



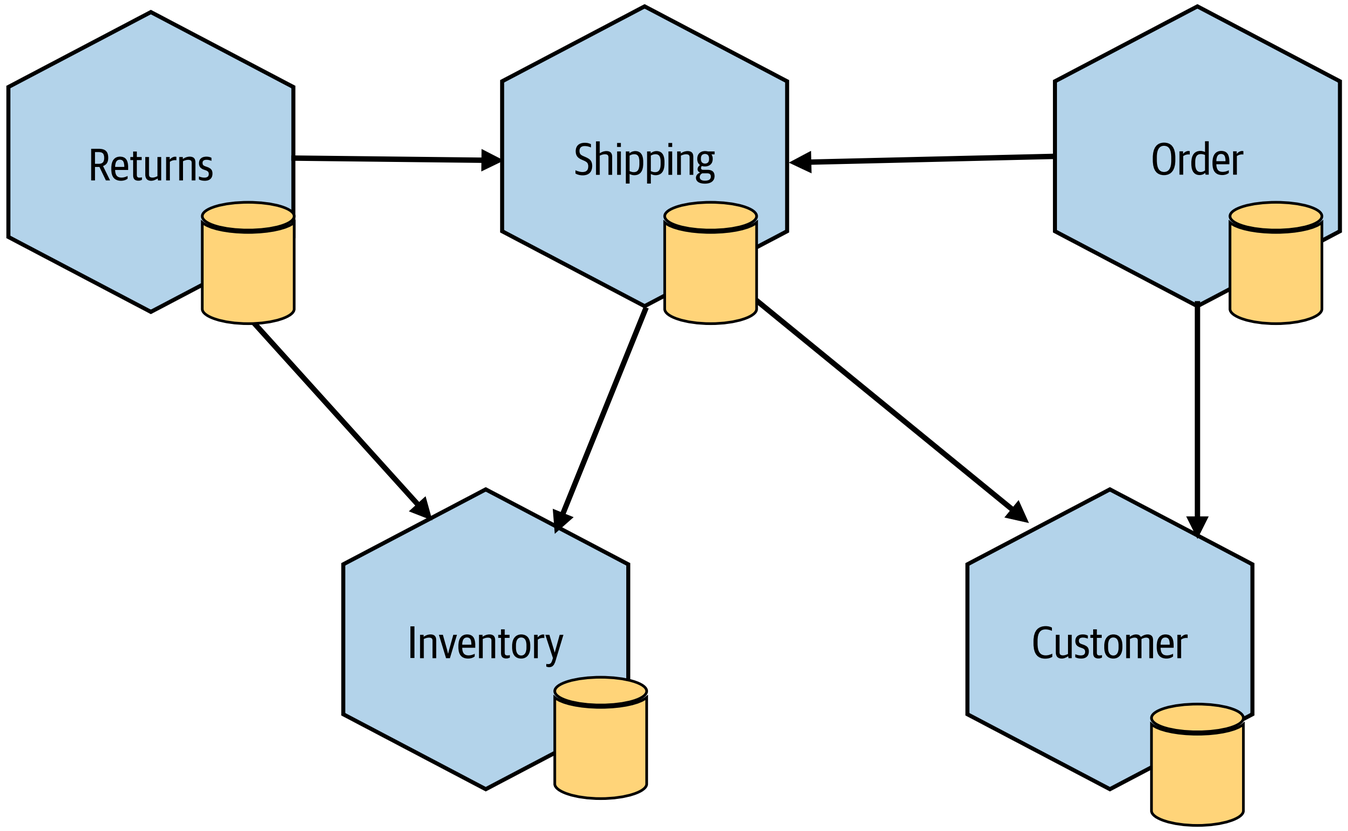
If microservices need to use a database to manage their state, this is hidden from other consumers

DATABASES



If microservices need to use a database to manage their state, this is hidden from other consumers

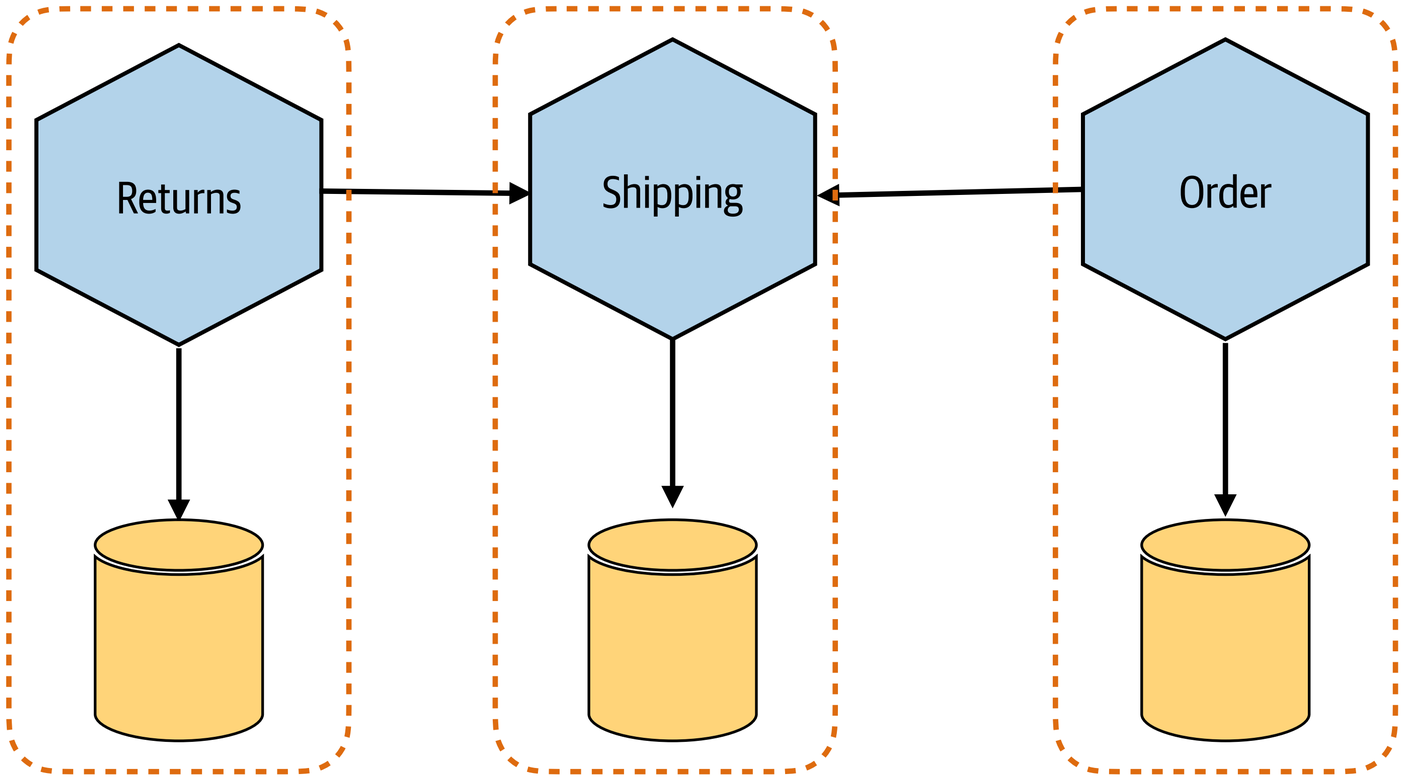
DATABASES



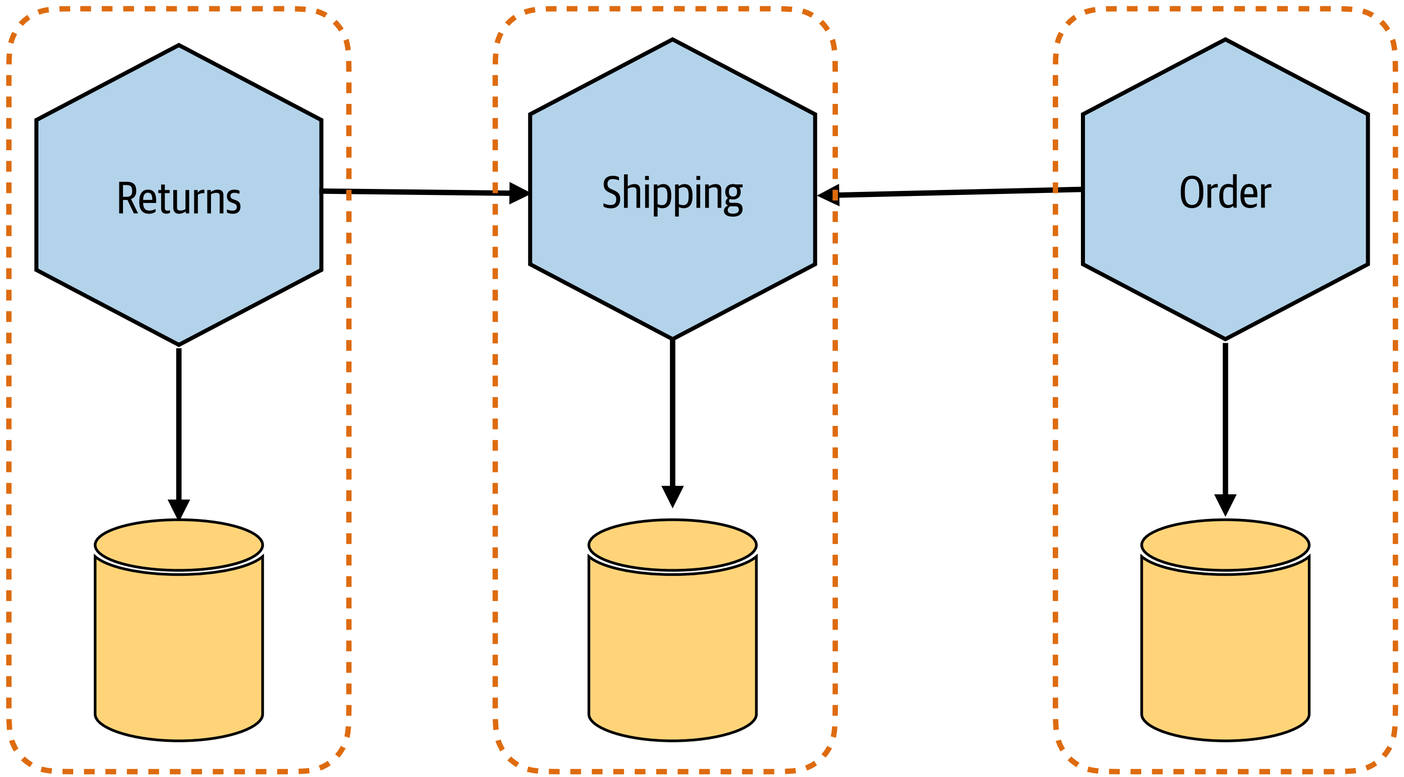
If microservices need to use a database to manage their state, this is hidden from other consumers

Does that mean lots of DB infrastructure to manage?

DATABASE HOSTING

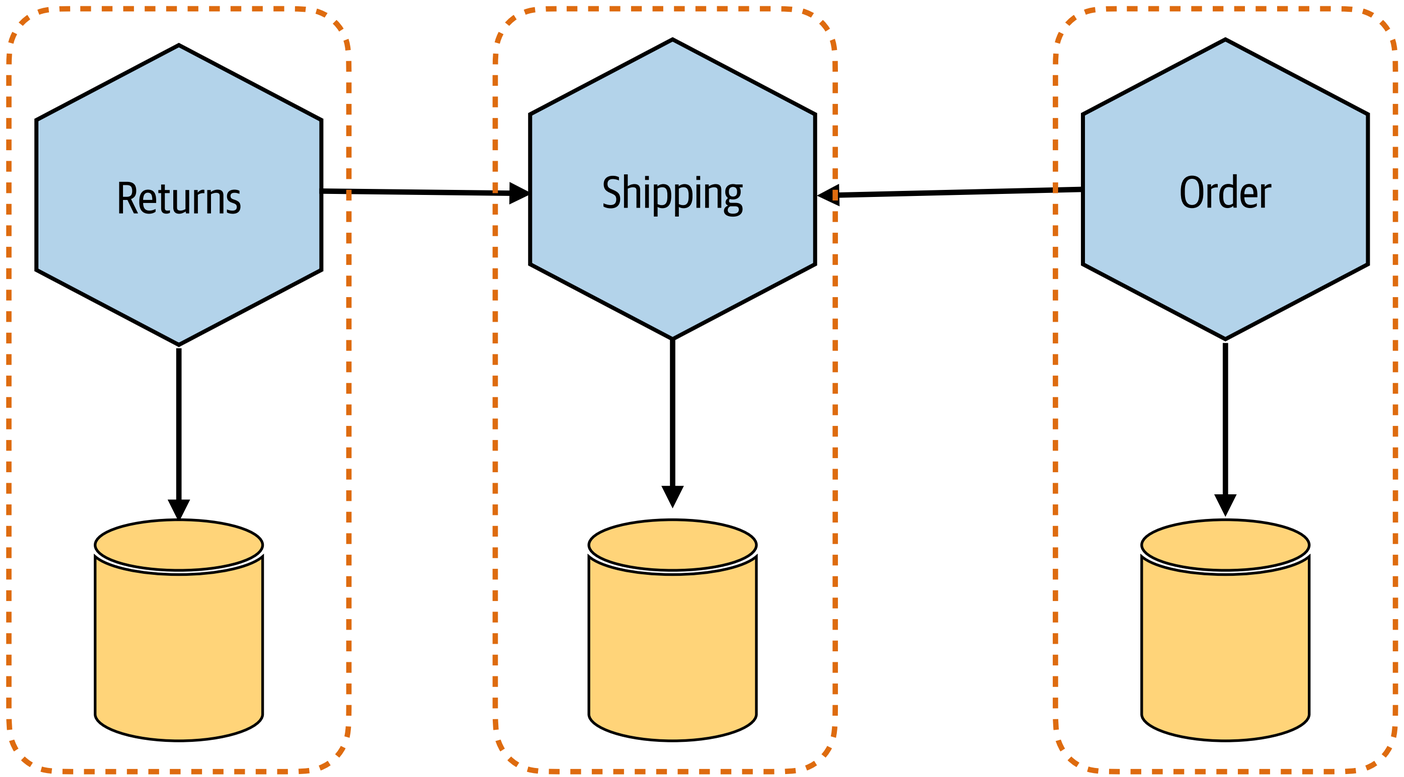


DATABASE HOSTING



We want each microservice's database to be logically isolated

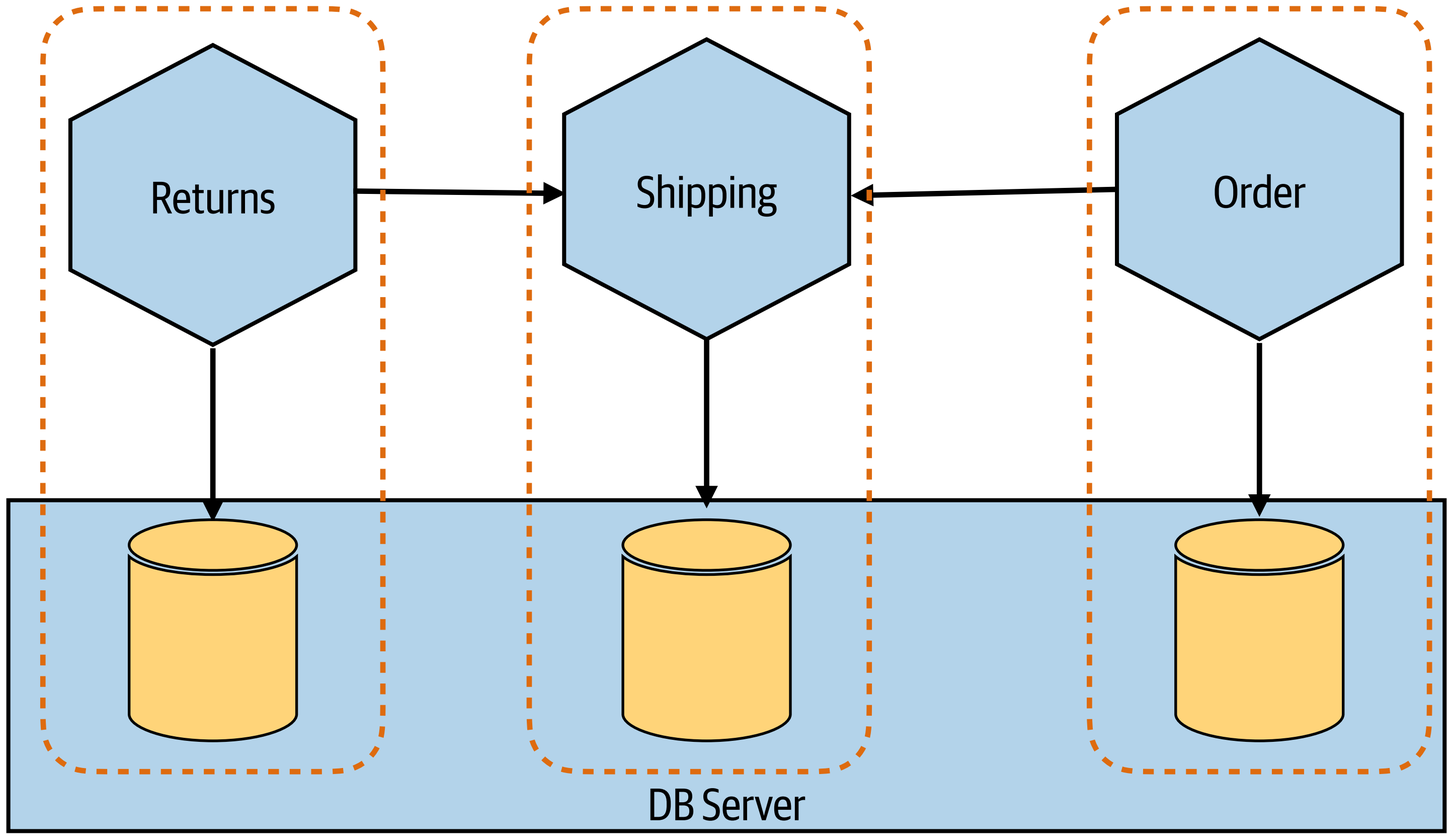
DATABASE HOSTING



We want each microservice's database to be logically isolated

But DB technology allows you to run multiple logical databases on the same infrastructure

DATABASE HOSTING



We want each microservice's database to be logically isolated

But DB technology allows you to run multiple logical databases on the same infrastructure

DEPLOYMENT OPTIONS

Bare Metal

DEPLOYMENT OPTIONS

Bare Metal

Virtual Machines

DEPLOYMENT OPTIONS

Bare Metal

Virtual Machines

Containers

DEPLOYMENT OPTIONS

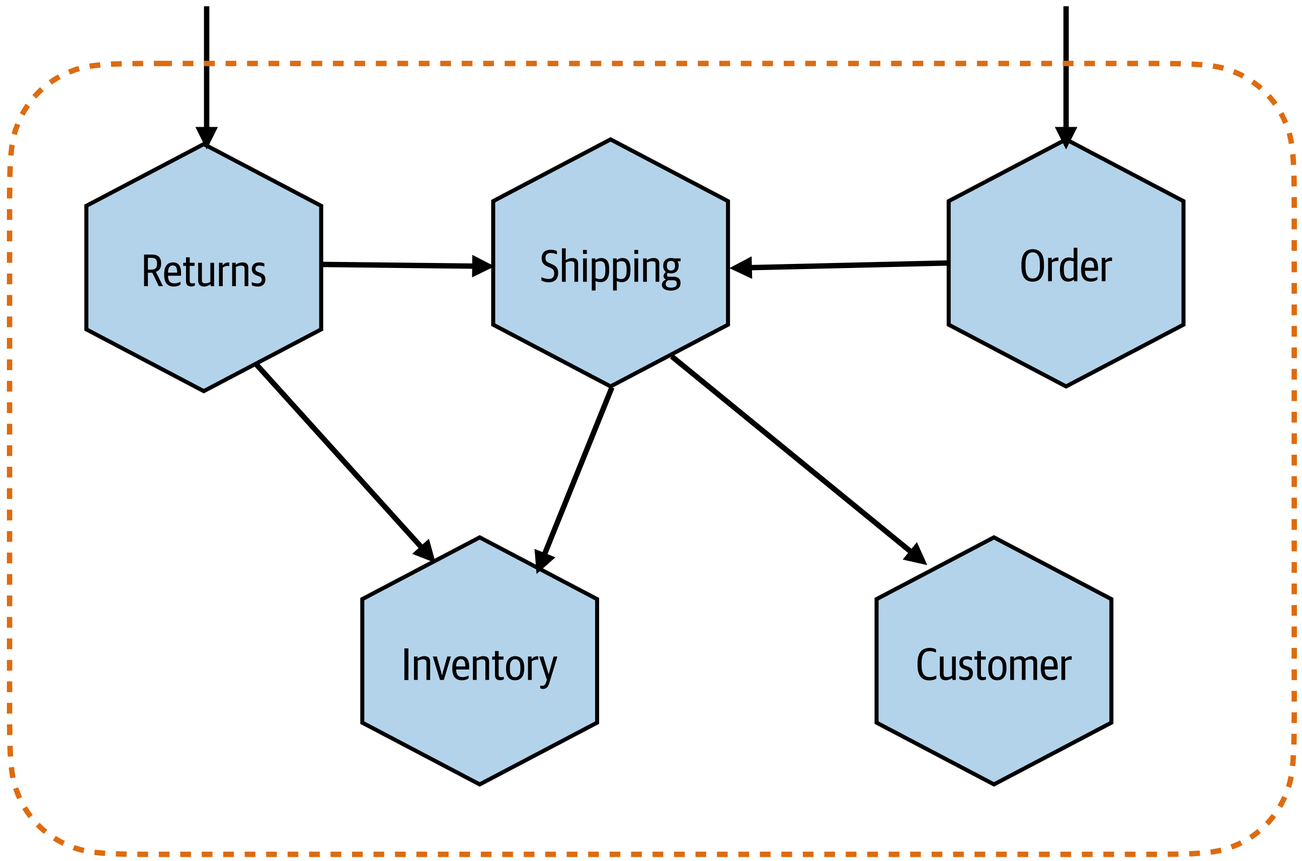
Bare Metal

Virtual Machines

Containers

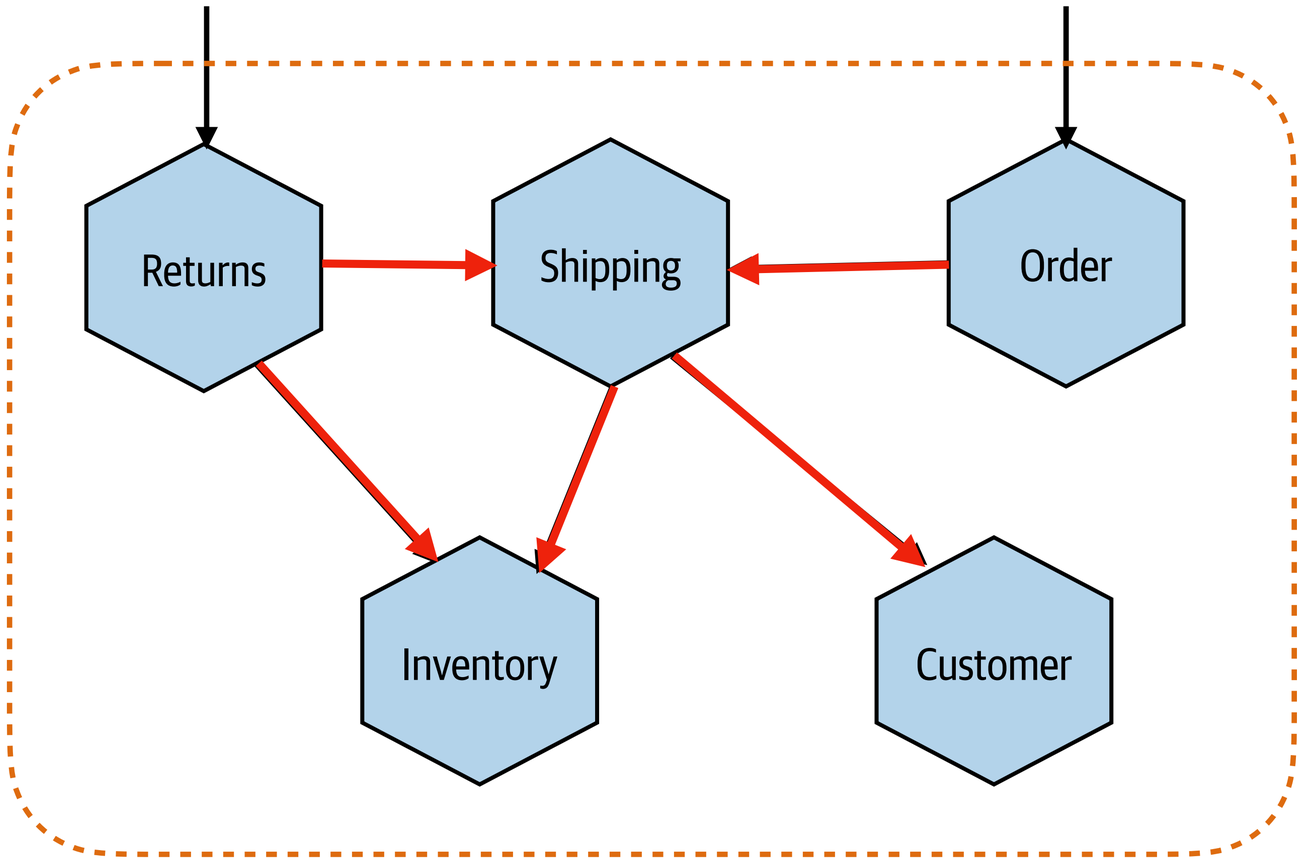
FAAS/PAAS

THE WIDER ECOSYSTEM



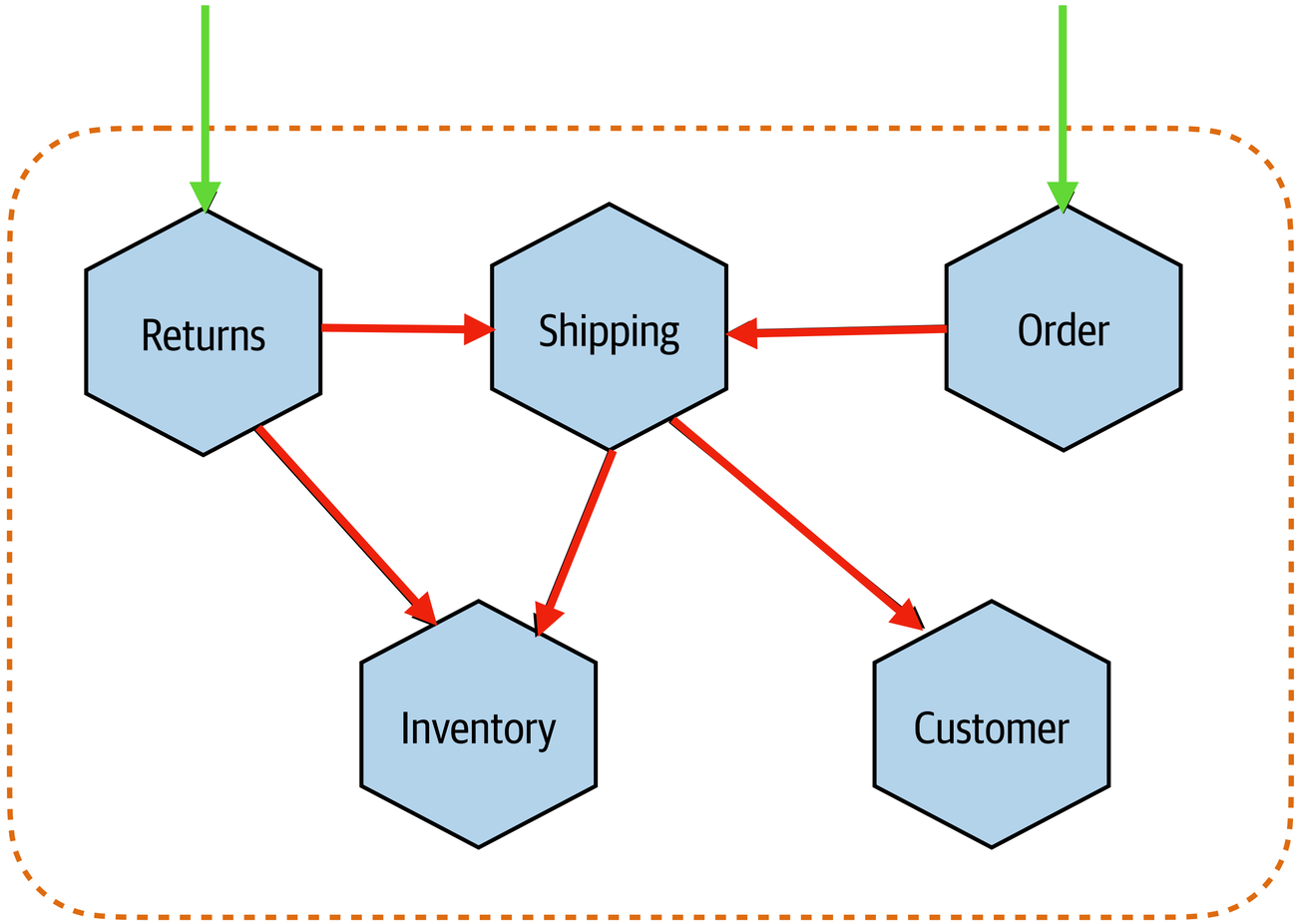
THE WIDER ECOSYSTEM

Comms within a perimeter - “east-west”



THE WIDER ECOSYSTEM

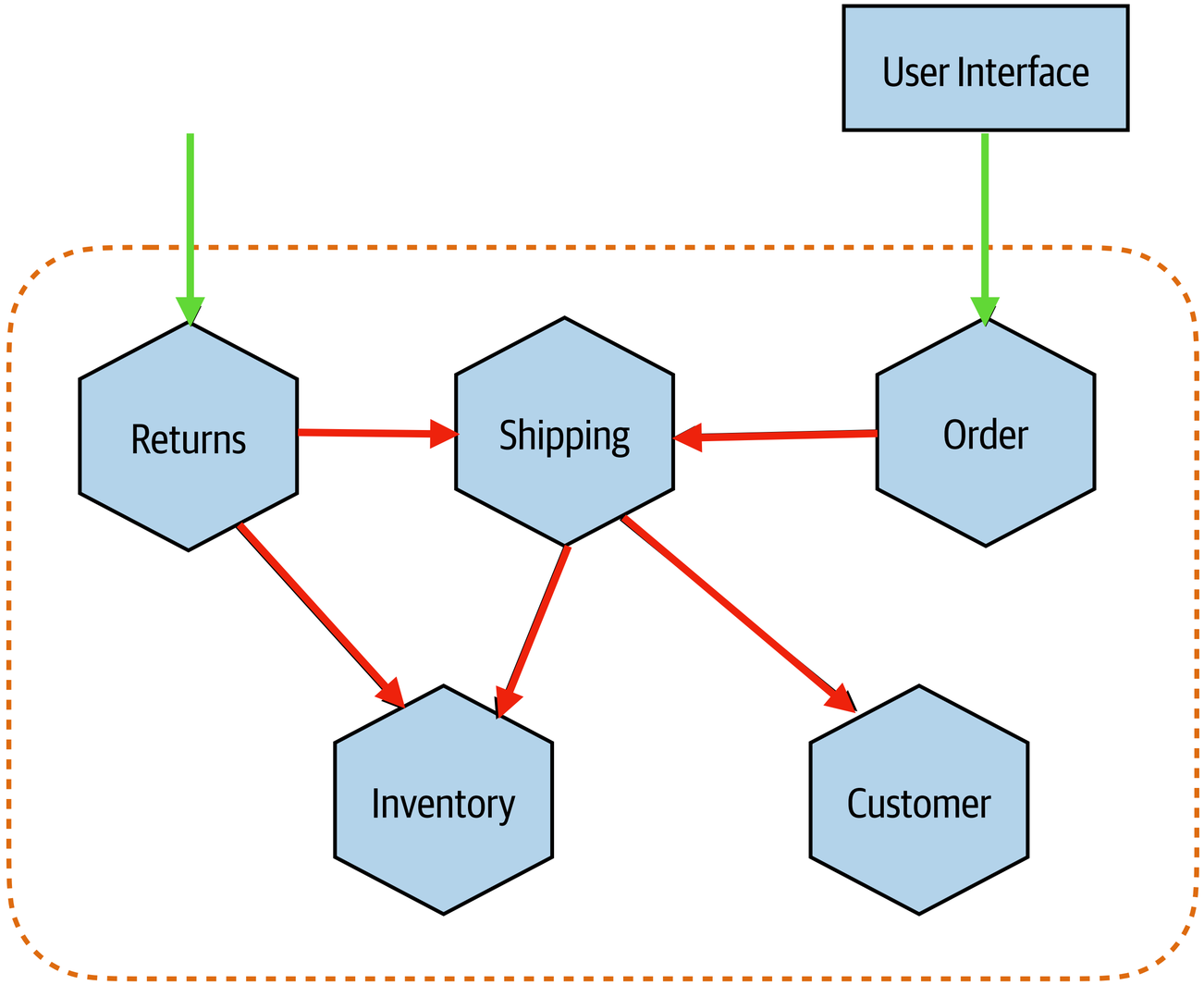
Comms within a perimeter - “east-west”



Comms in/out of the perimeter - “north-south”

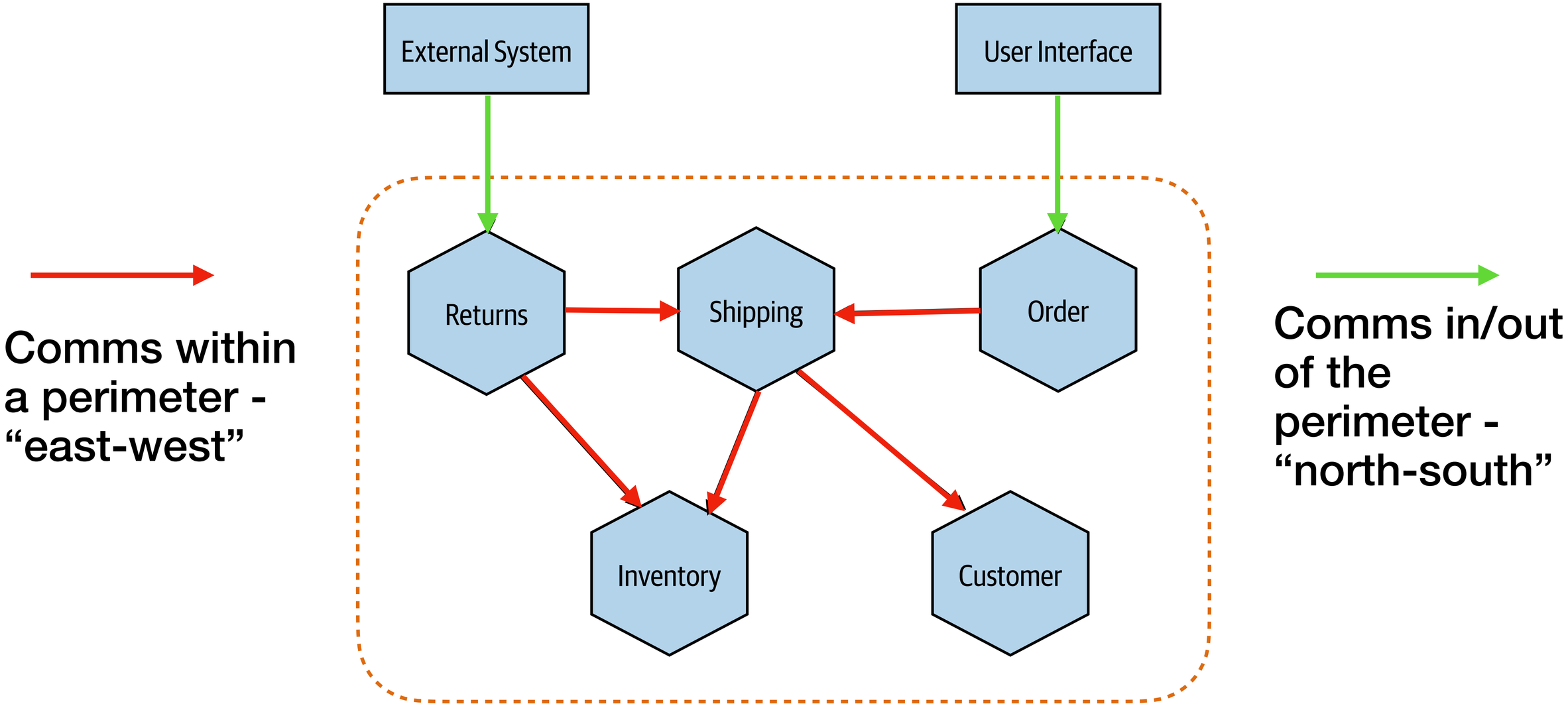
THE WIDER ECOSYSTEM

Comms within a perimeter - “east-west”

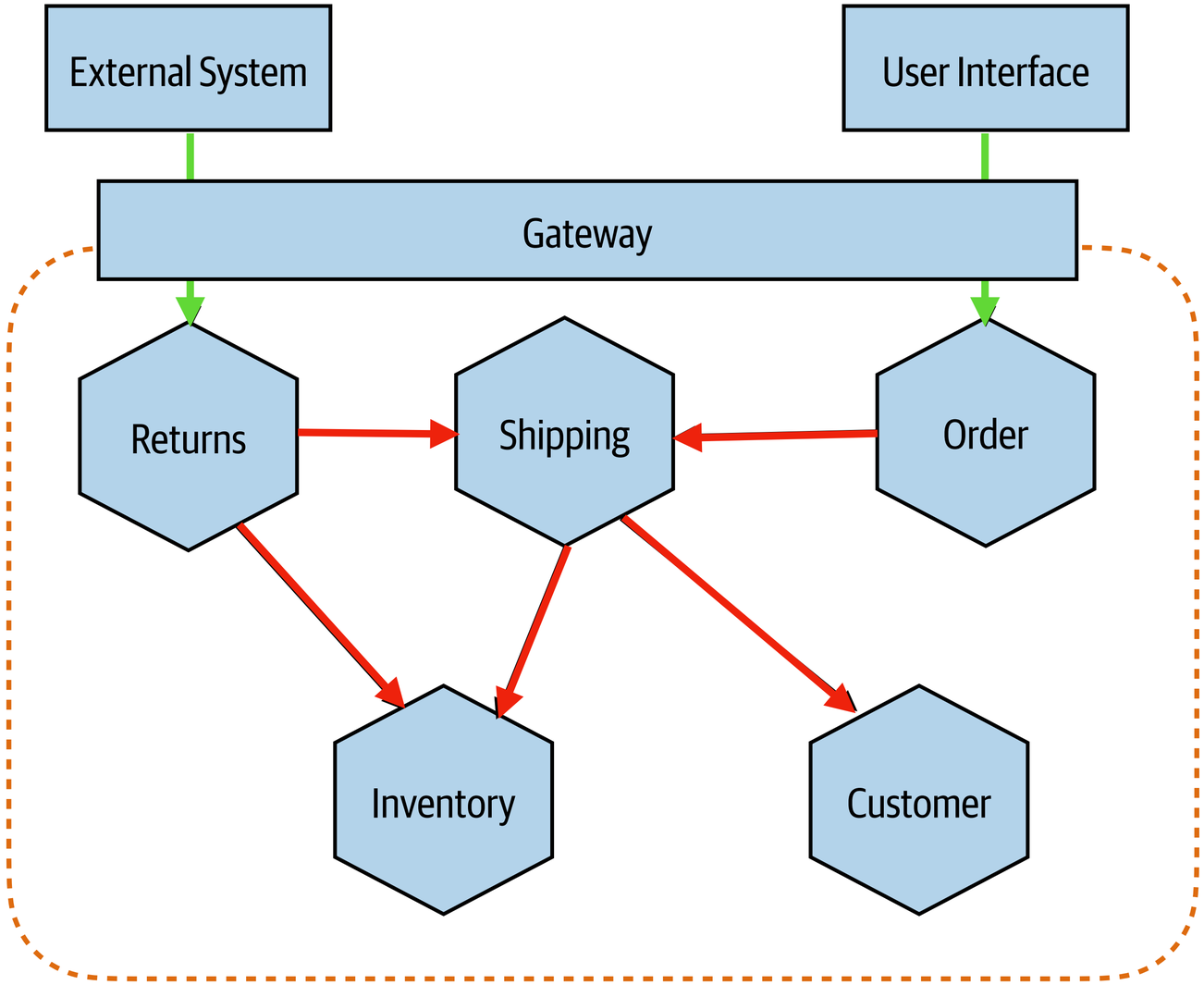


Comms in/out of the perimeter - “north-south”

THE WIDER ECOSYSTEM



THE WIDER ECOSYSTEM



Comms within a perimeter - “east-west”

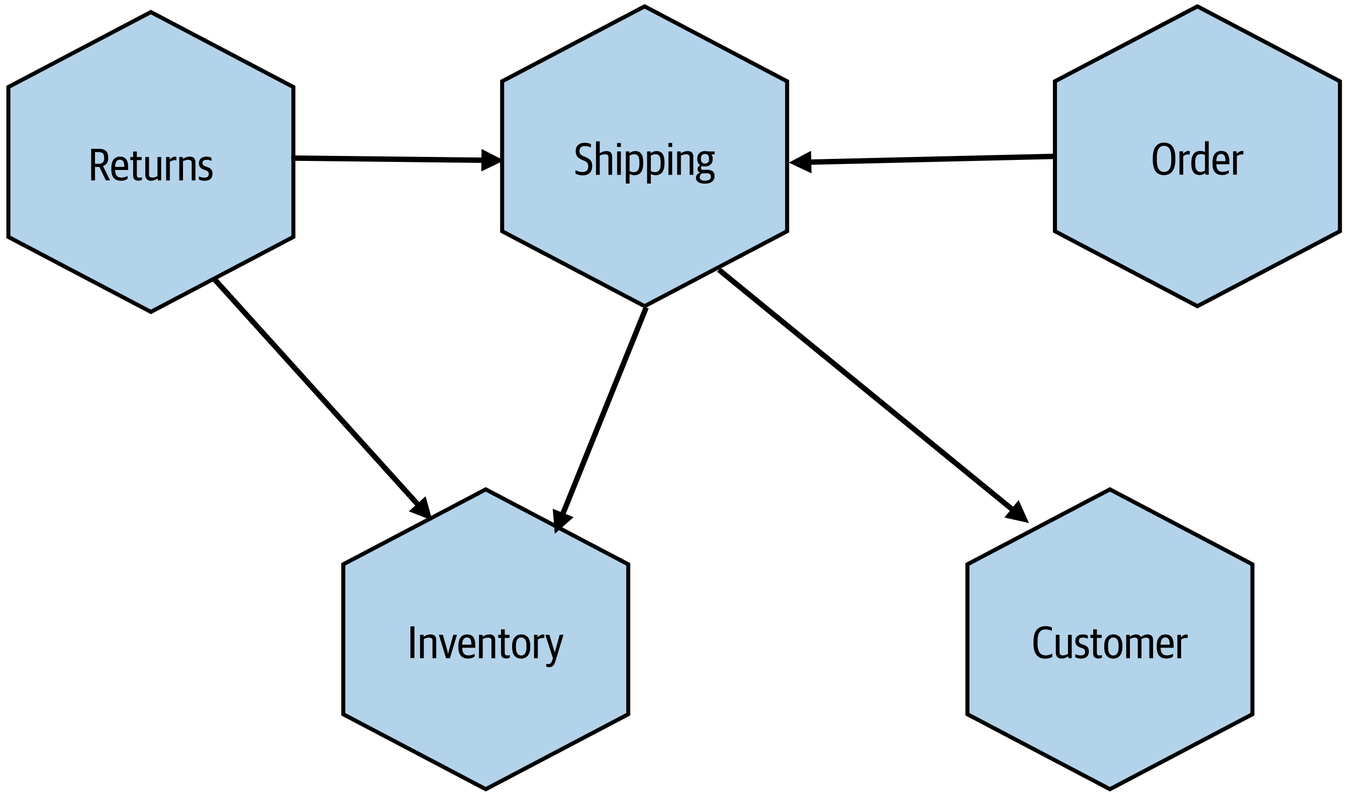
Comms in/out of the perimeter - “north-south”

Technology

SOA

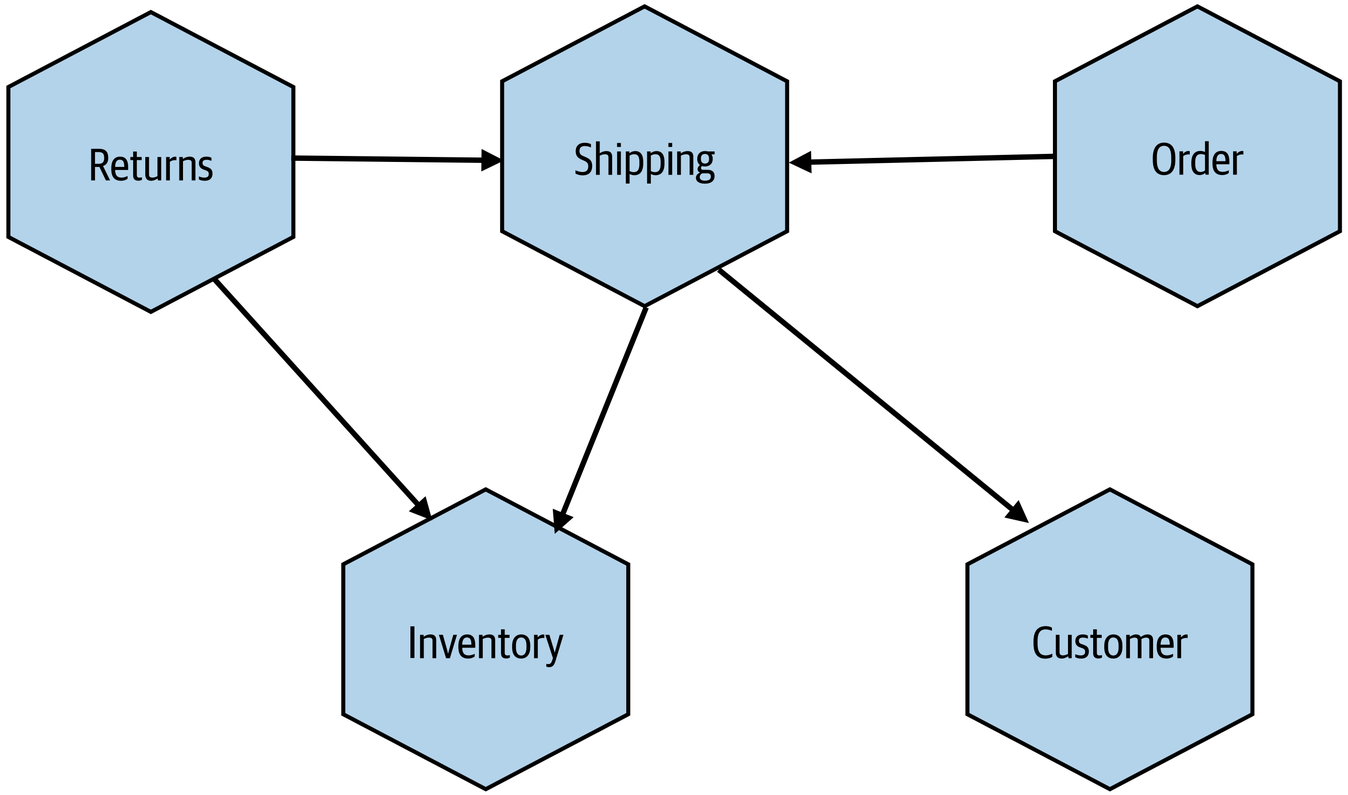
Size

BUSINESS DOMAIN?



Makes alignment with org structure easier

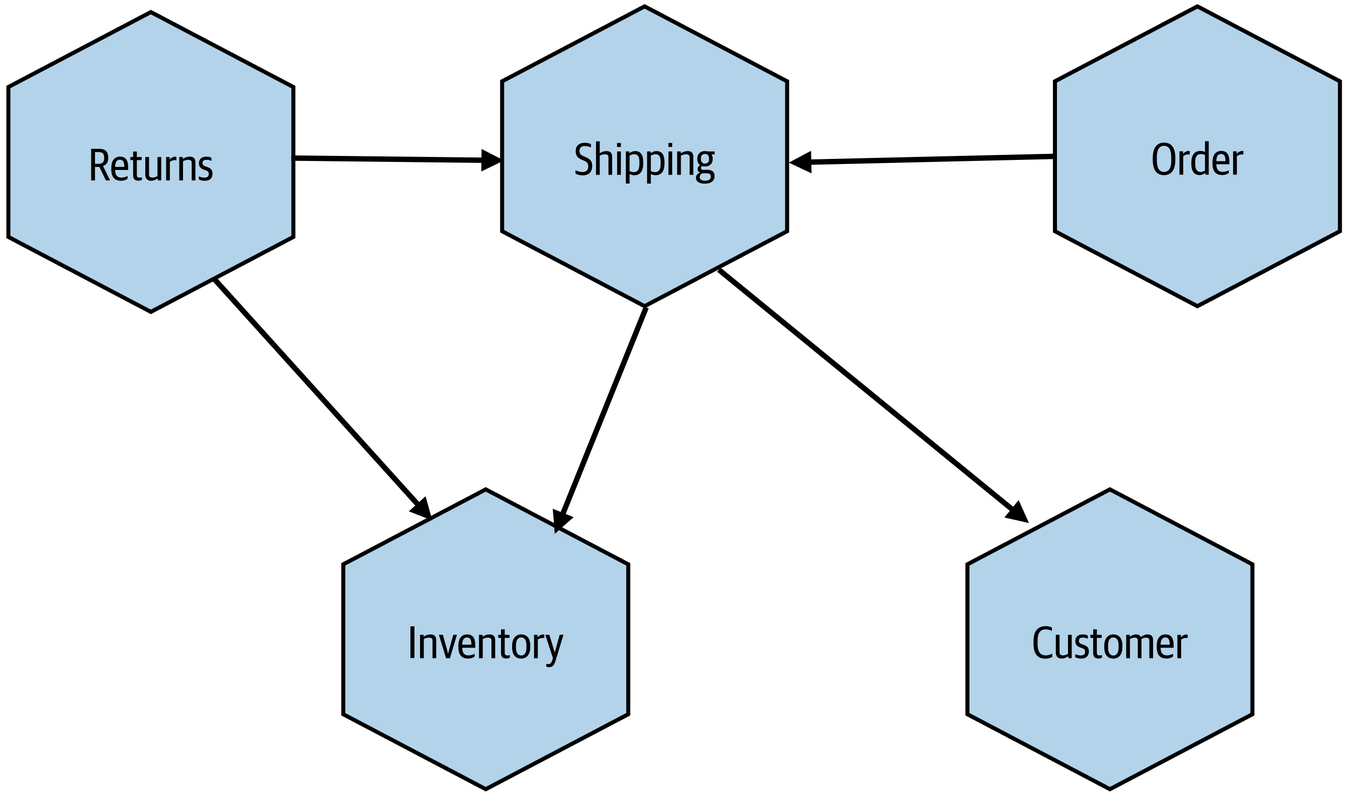
BUSINESS DOMAIN?



Makes alignment with org structure easier

Domain boundaries tend to be more “stable”

BUSINESS DOMAIN?



Makes alignment with org structure easier

Domain boundaries tend to be more “stable”

Can recombine functionality more easily to deliver new experiences

2/4 Independent Deployability

Independent Deployability

The ability to make a change to a microservice, deploy that microservice into production without having to change or deploy any other microservices

POLL: CAN YOU DEPLOY INDEPENDENTLY?

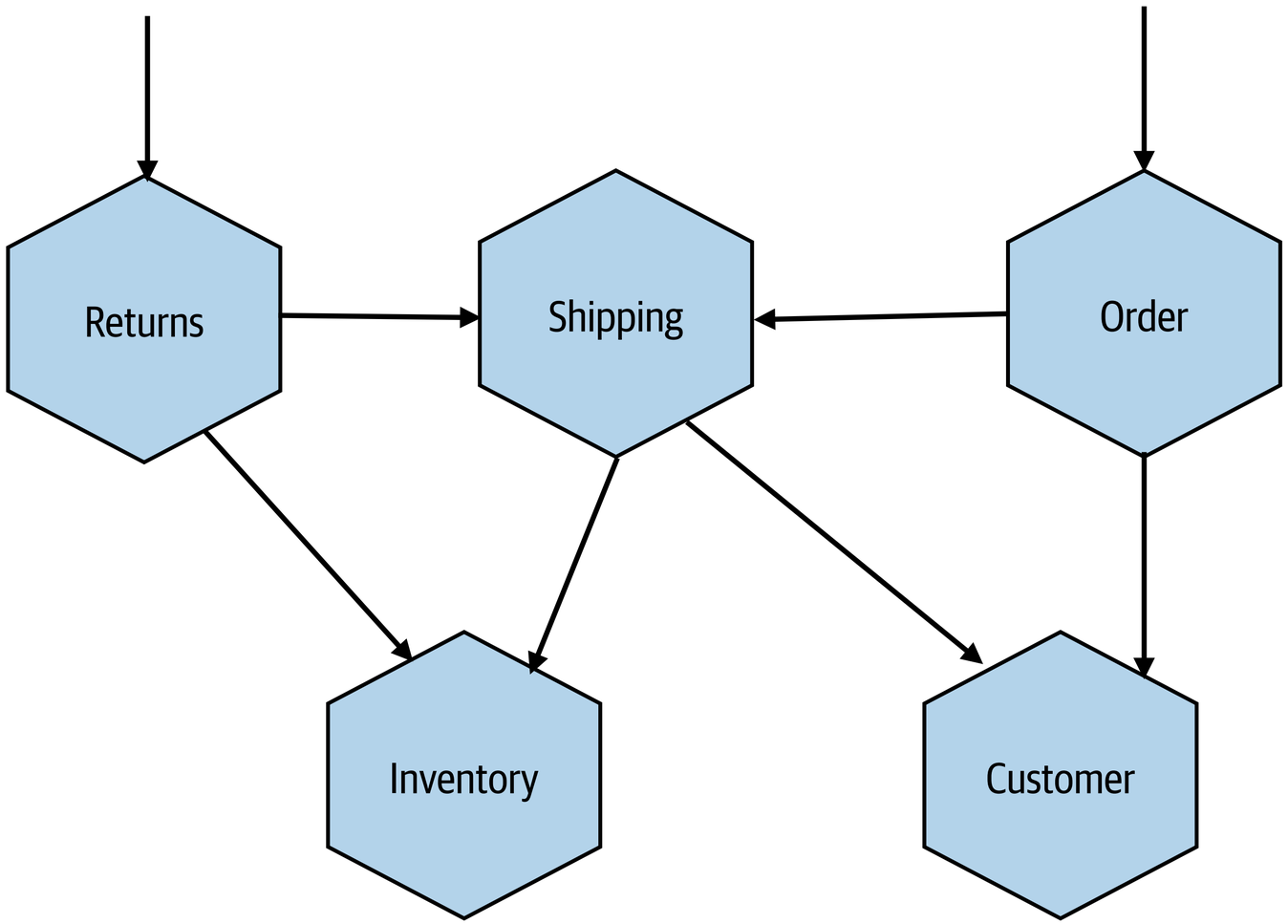
NA (we don’t have microservices)

Yes! Every microservice deployment is done separately

Sometimes….

No - we deploy everything together!

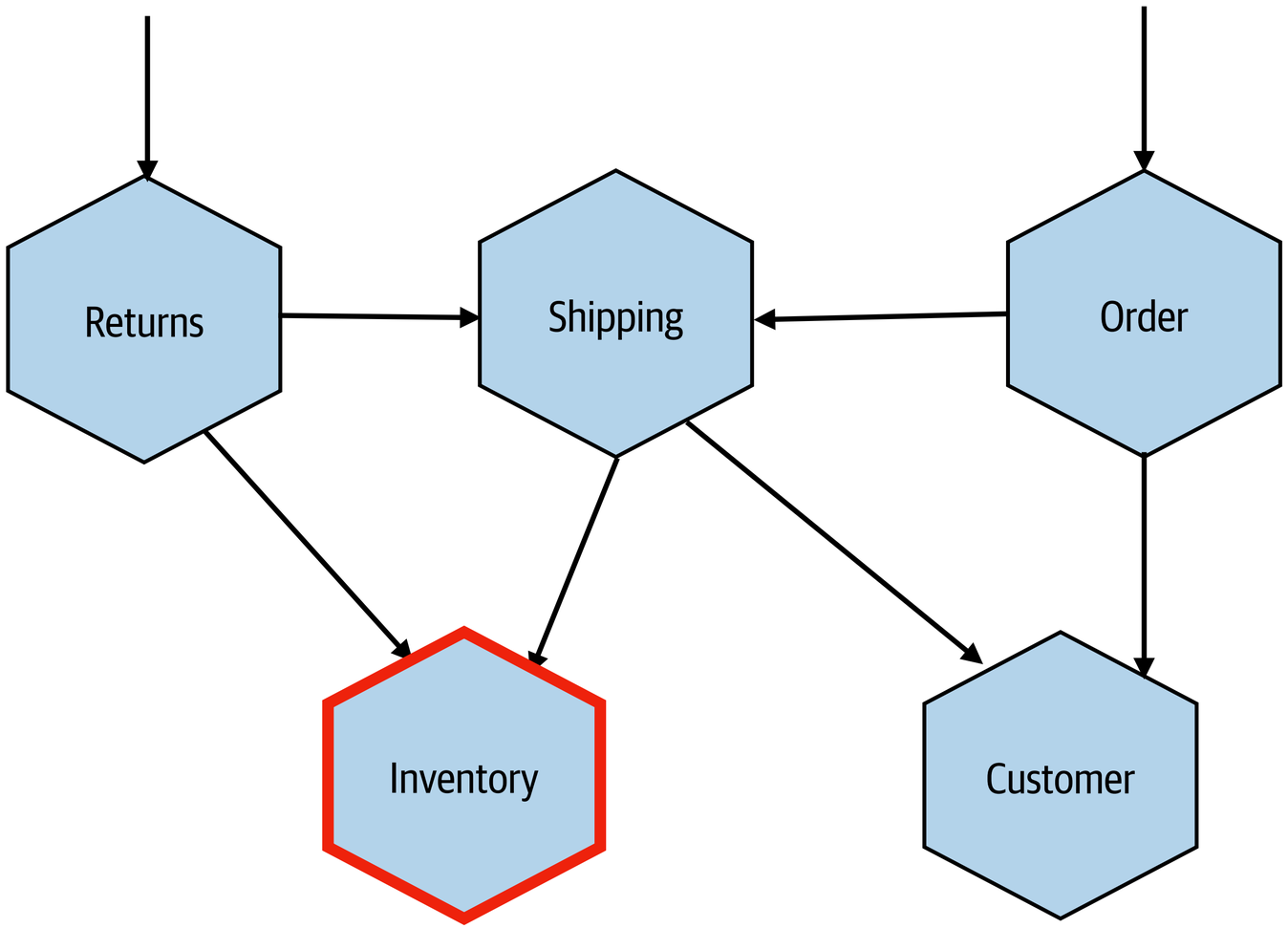
INDEPENDENT DEPLOYABILITY



When we make a change, we need to make sure we don't break upstream consumers

So backwards compatibility is key

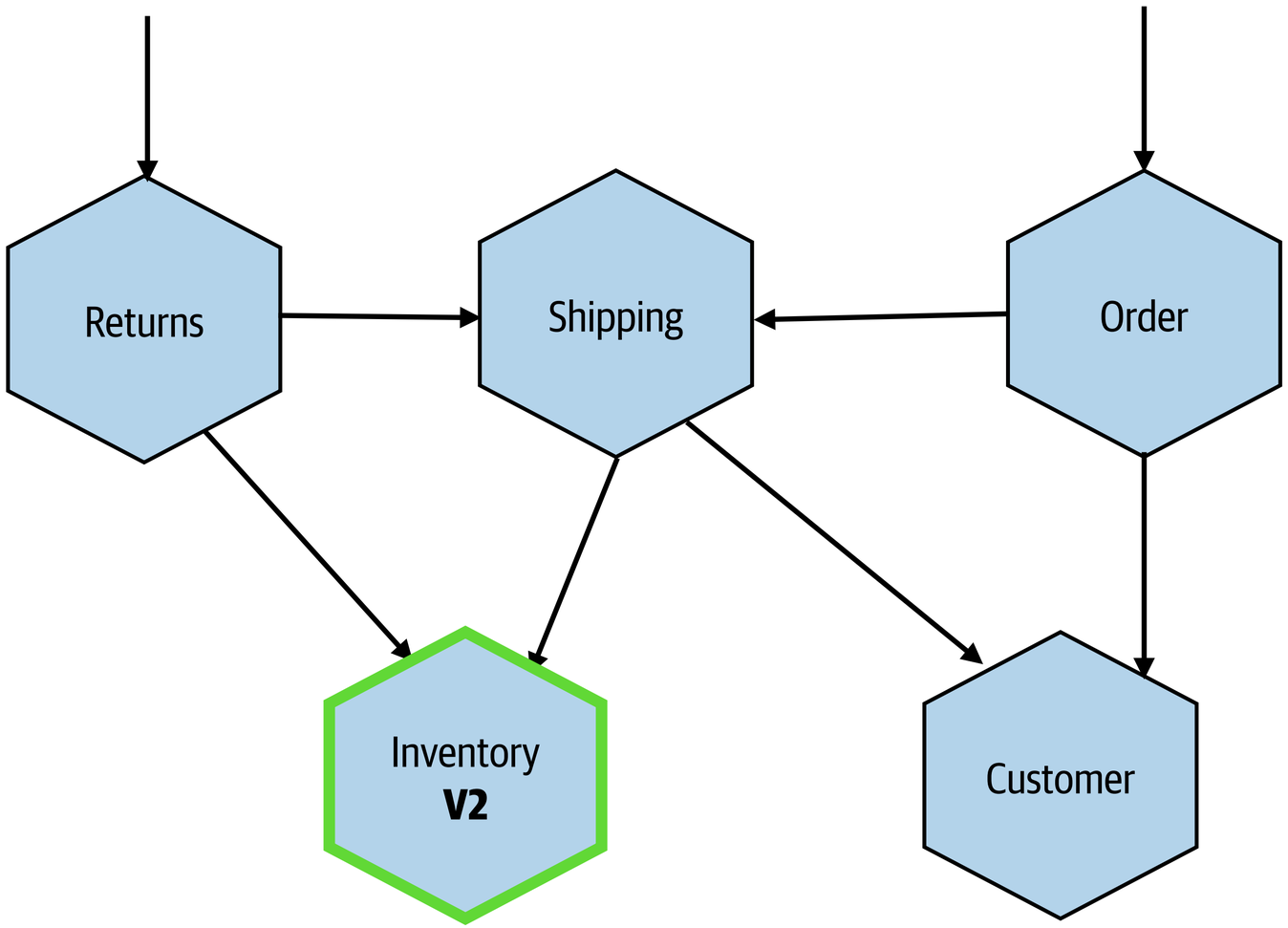
INDEPENDENT DEPLOYABILITY



When we make a change, we need to make sure we don't break upstream consumers

So backwards compatibility is key

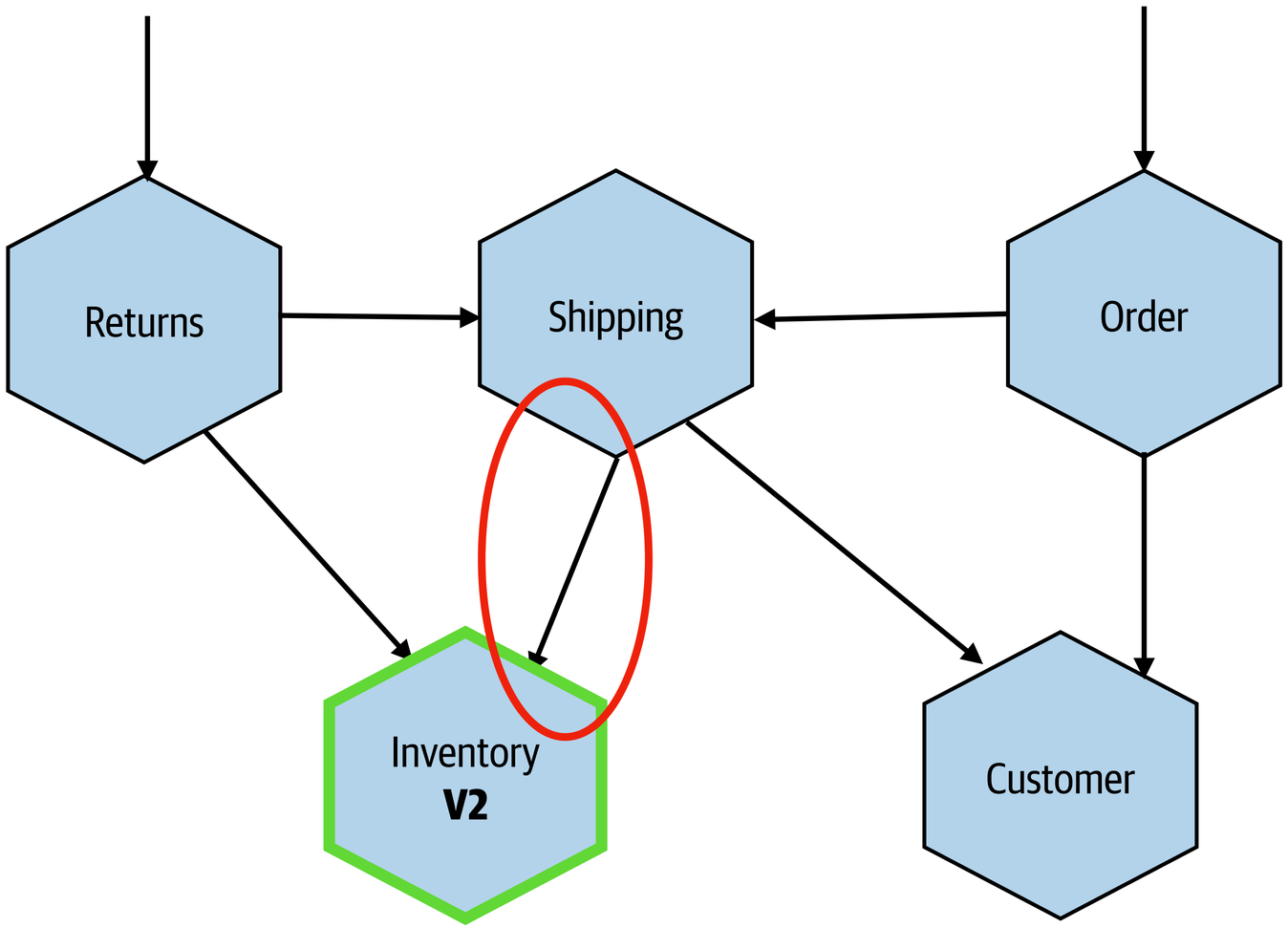
INDEPENDENT DEPLOYABILITY



When we make a change, we need to make sure we don't break upstream consumers

So backwards compatibility is key

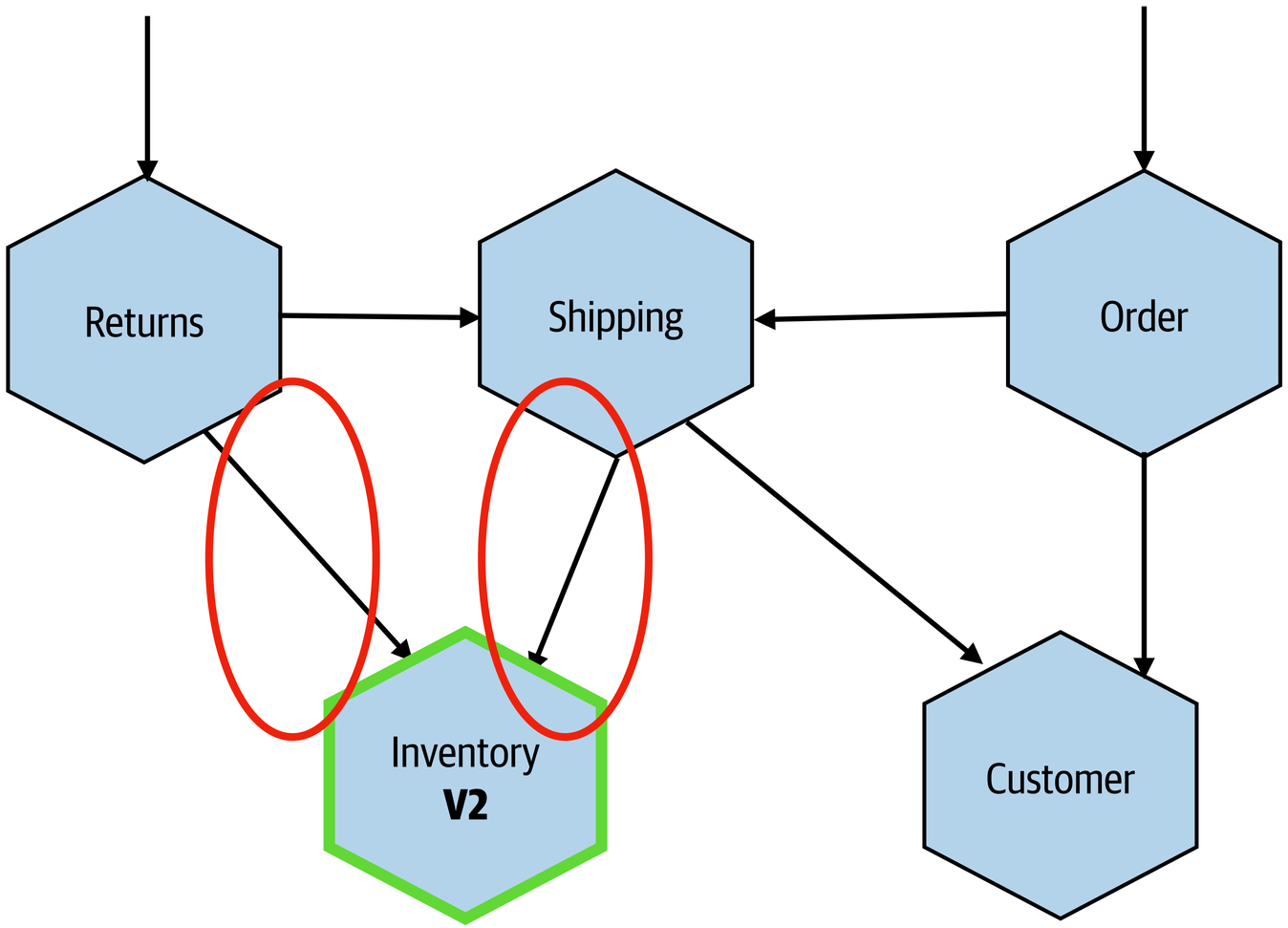
INDEPENDENT DEPLOYABILITY



When we make a change, we need to make sure we don't break upstream consumers

So backwards compatibility is key

INDEPENDENT DEPLOYABILITY



When we make a change, we need to make sure we don't break upstream consumers

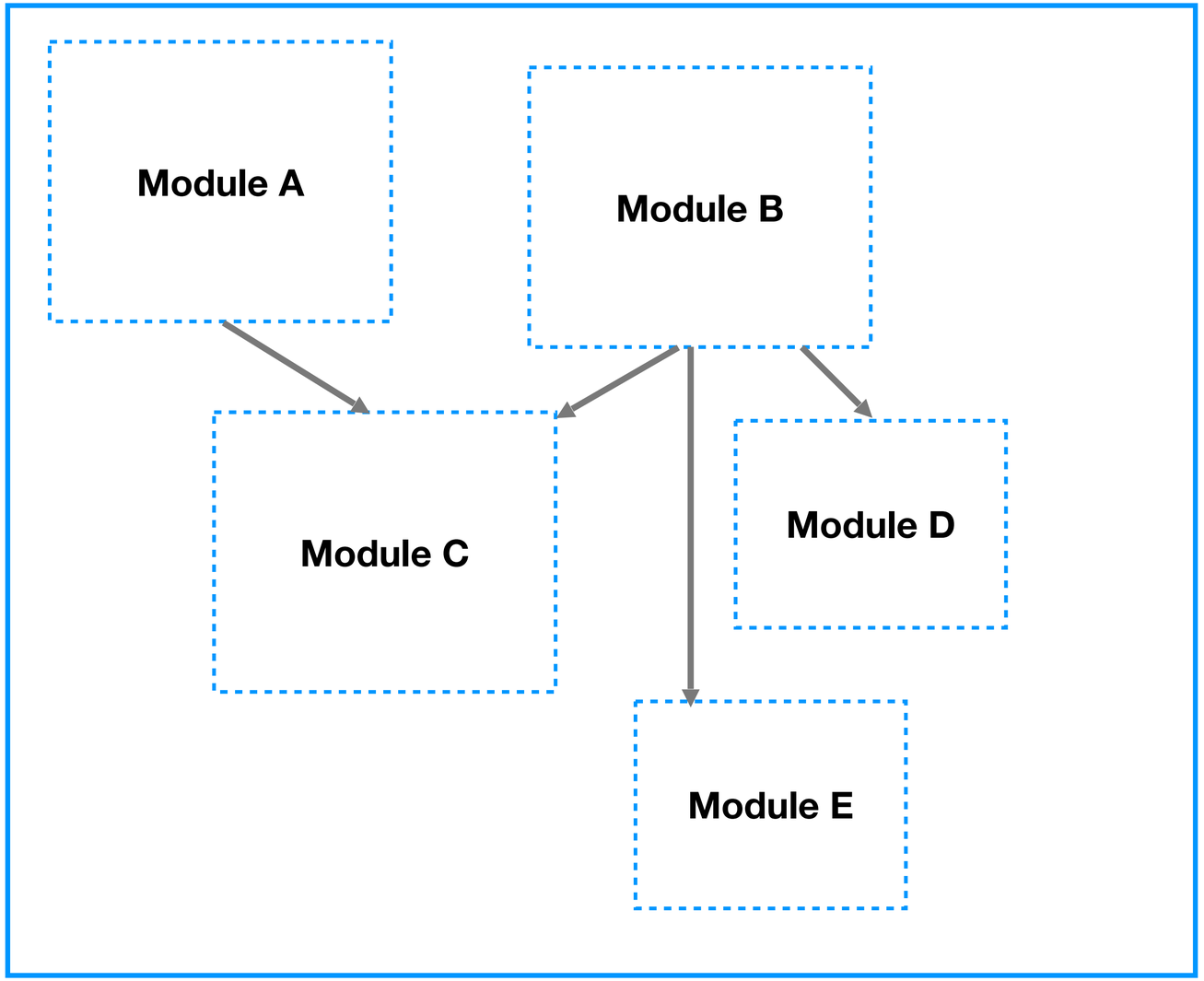
So backwards compatibility is key

So how do we get good at not breaking consumers?

https://www.flickr.com/photos/photographingtravis/16939917735/

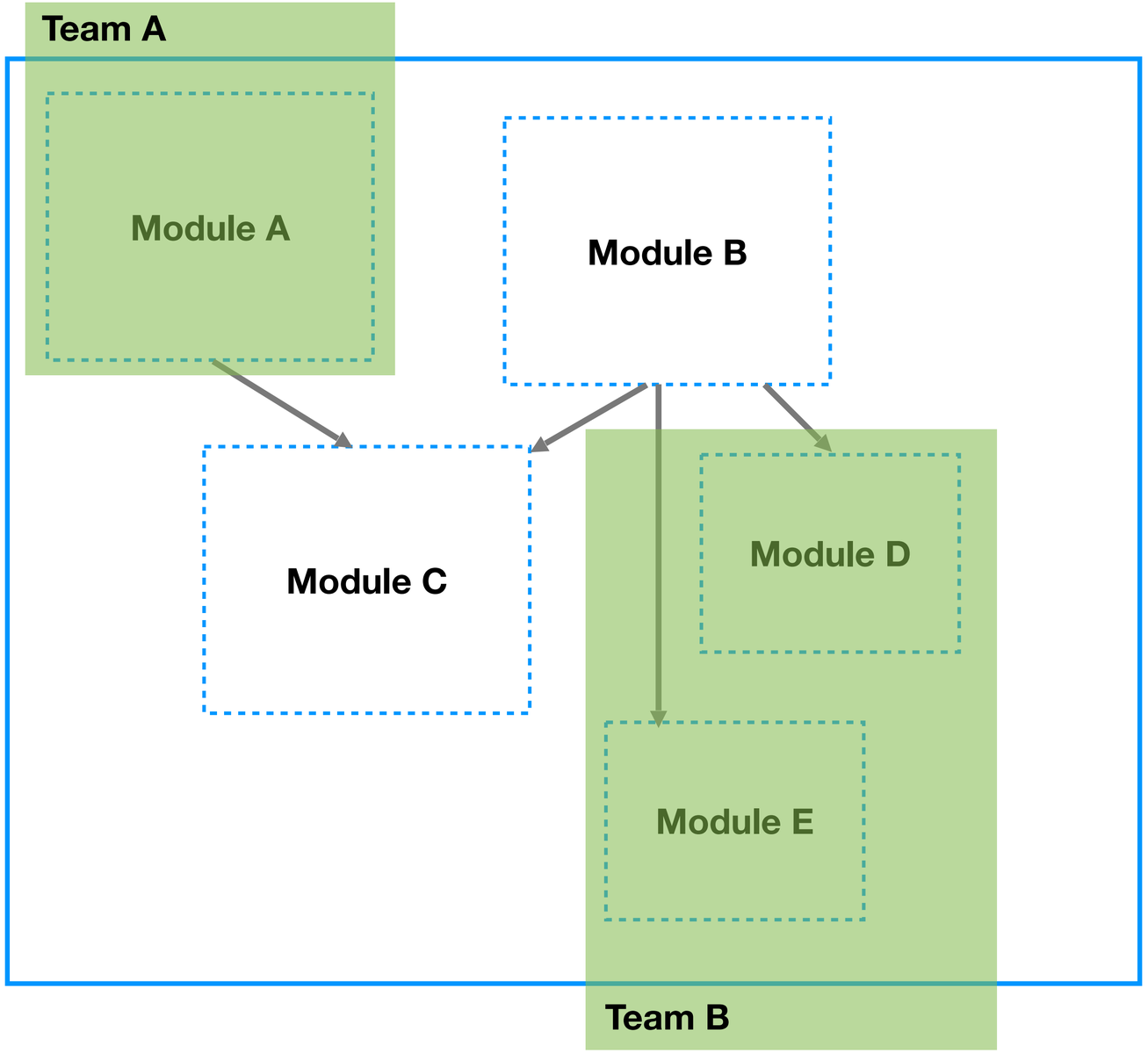
MODULES

MODULES



MODULES

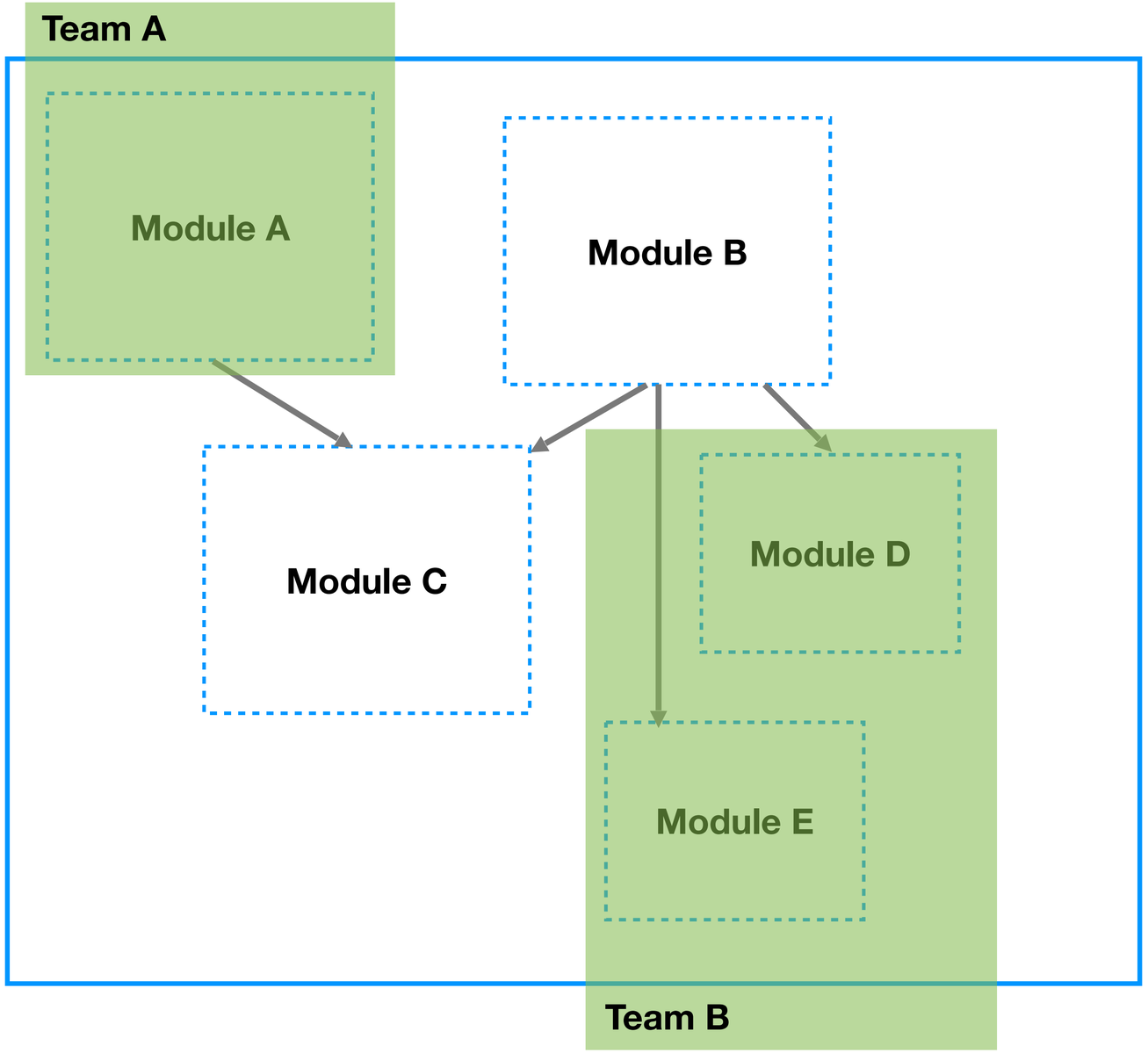
Allow for independent working



MODULES

Allow for independent working

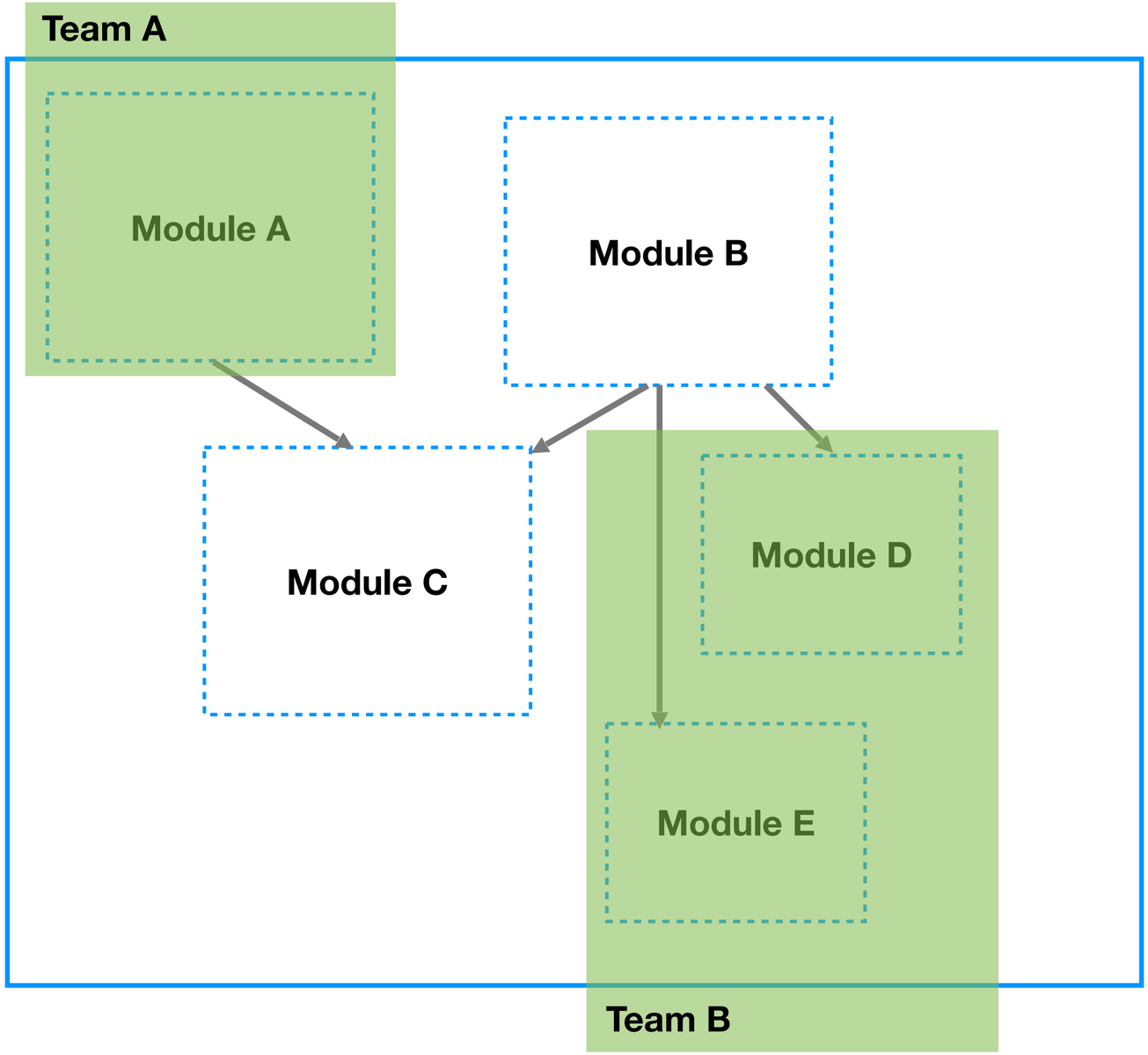
And reuse



MODULES

Allow for independent working

And reuse

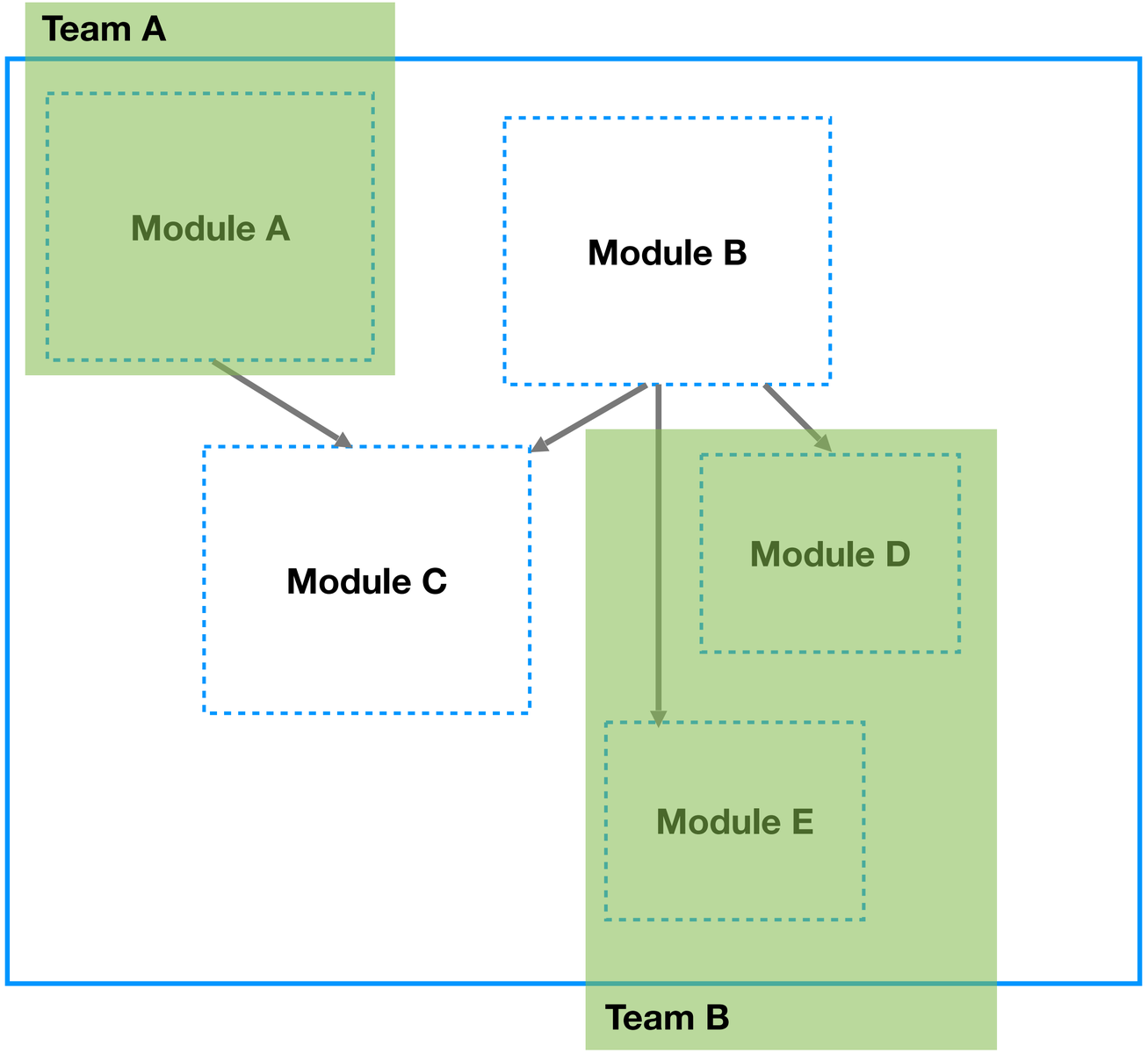


Namespaces, packages

MODULES

Allow for independent working

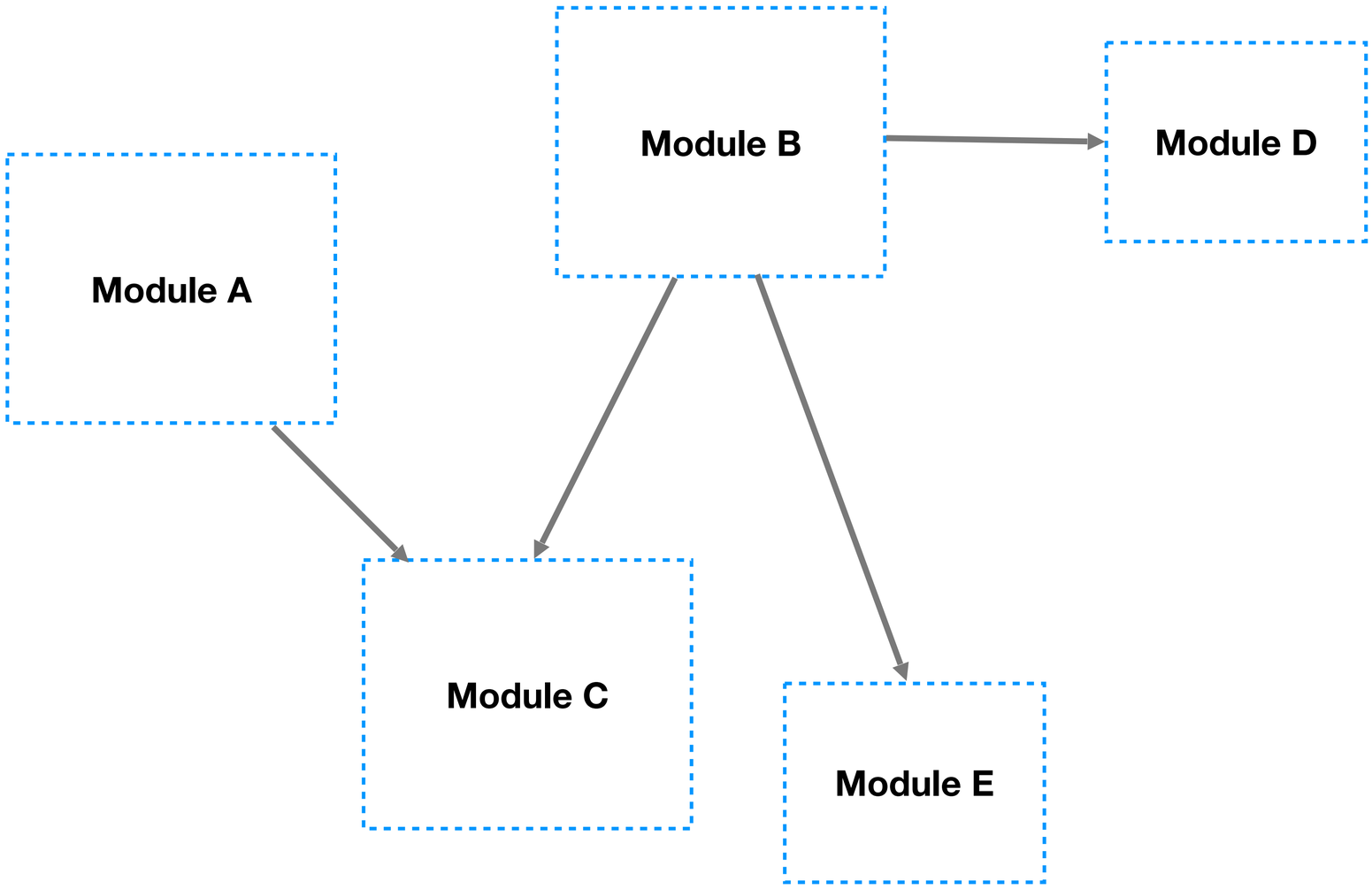
And reuse



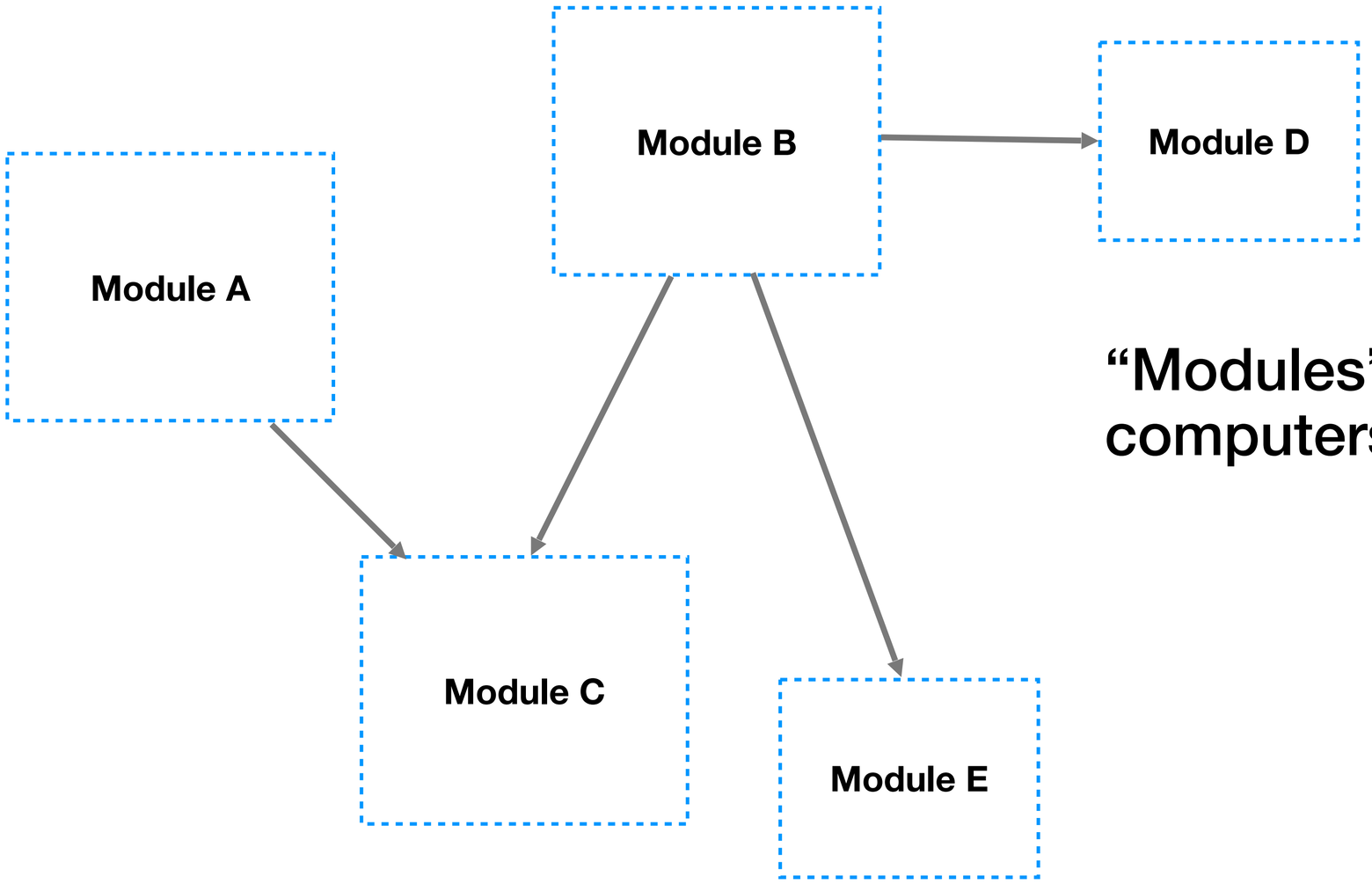
Namespaces, packages

NPMs, Gems, JARs etc

AND MICROSERVICES?

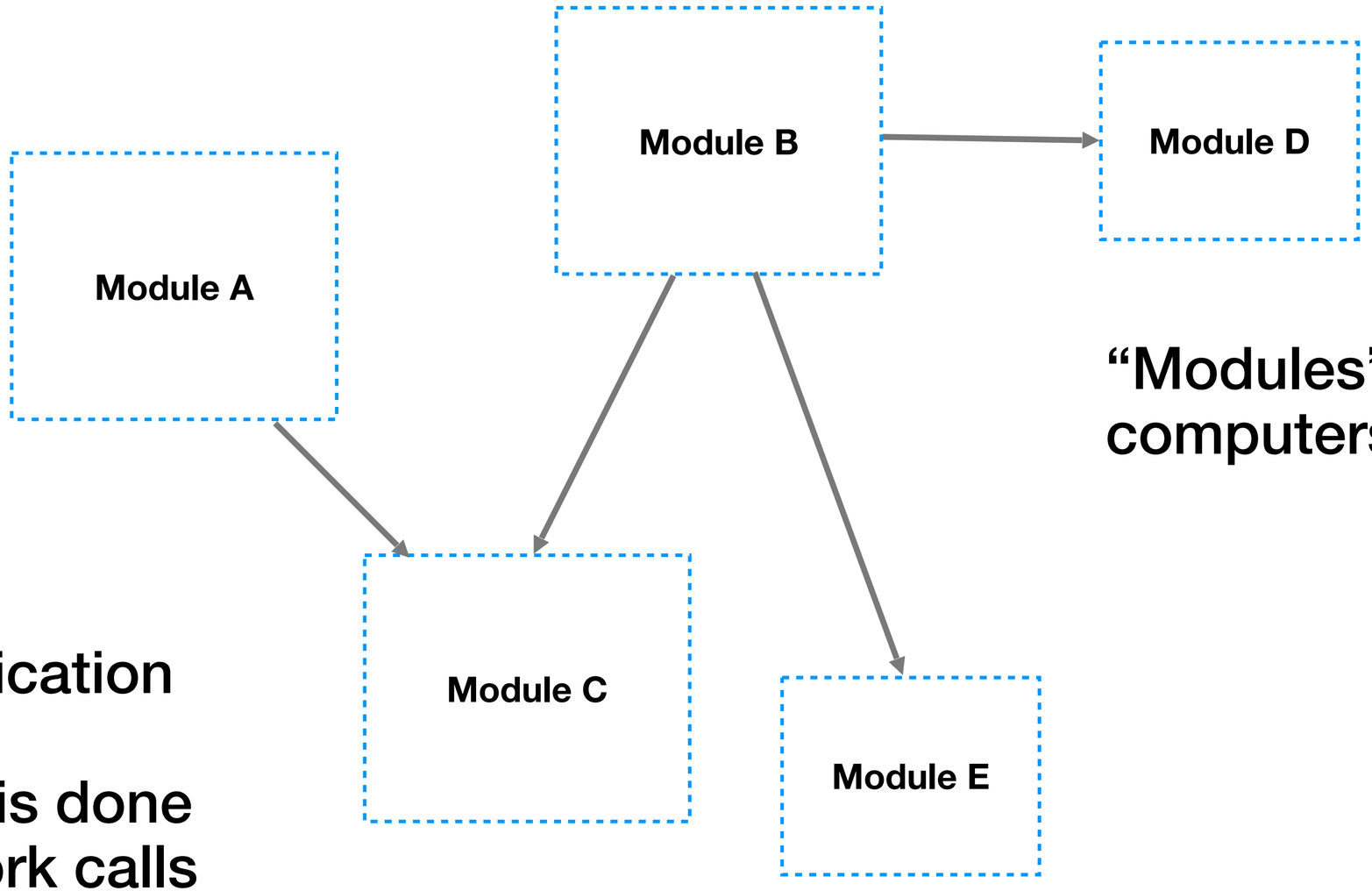


AND MICROSERVICES?



“Modules” run on different computers

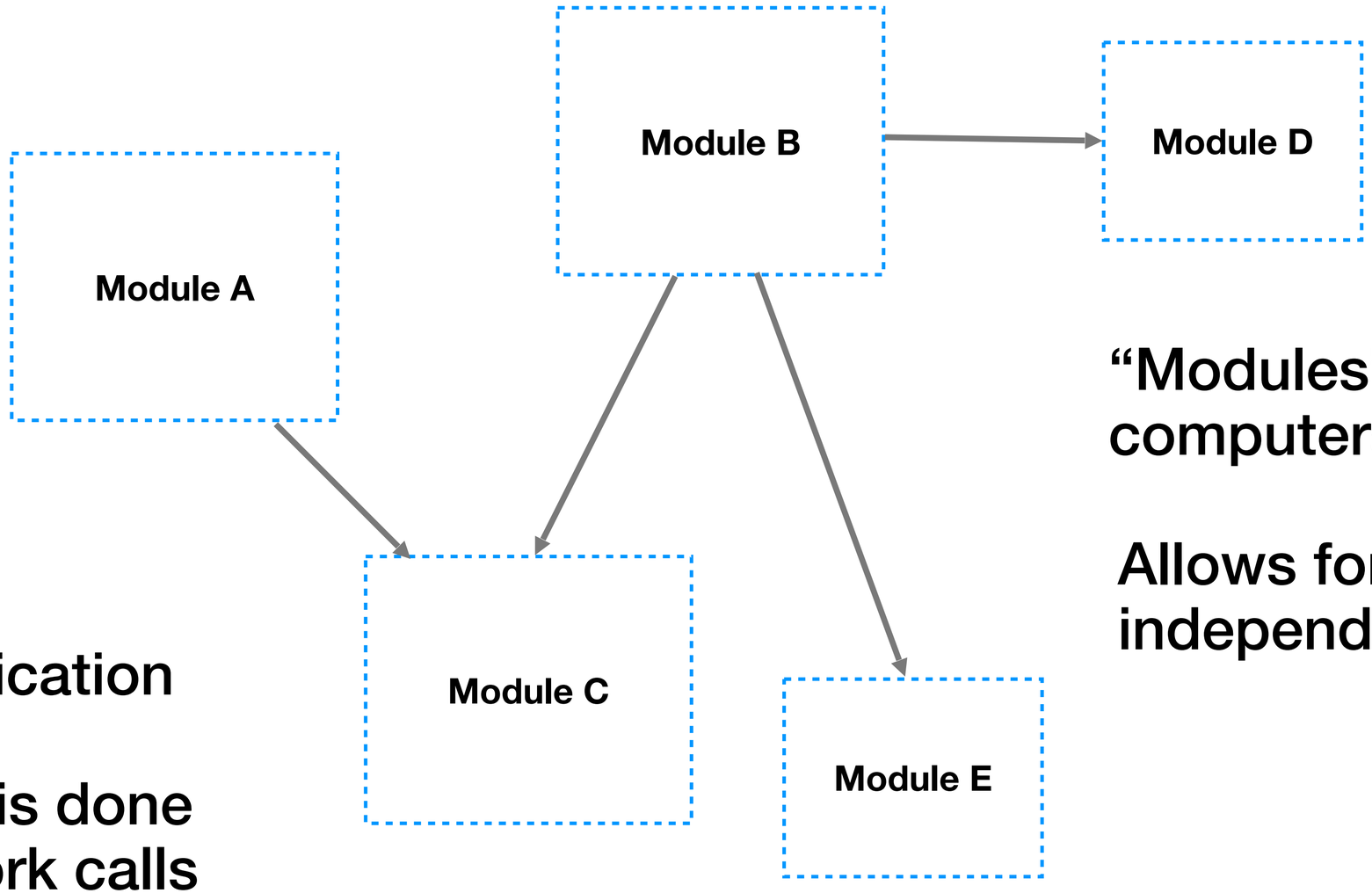
AND MICROSERVICES?



“Modules” run on different computers

Communication between modules is done via network calls

AND MICROSERVICES?



"Modules" run on different computers

Allows for easier independent deployability

Communication between modules is done via network calls

Microservices, when done right, are a form of modular architecture

https://www.flickr.com/photos/ibm_media/33838065805/

WHAT MAKES A GOOD MODULE? (ACCORDING TO PARNAS)

Improved Development Time

Allow for work to be done in parallel

WHAT MAKES A GOOD MODULE? (ACCORDING TO PARNAS)

Improved Development Time

Allow for work to be done in parallel

Comprehensability

Can understand how a module works (in isolation)

WHAT MAKES A GOOD MODULE? (ACCORDING TO PARNAS)

Improved Development Time

Allow for work to be done in parallel

Comprehensability

Can understand how a module works (in isolation)

Flexibility

Can be changed independently from each other

WHAT MAKES A GOOD MODULE? (ACCORDING TO PARNAS)

Improved Development Time
Allow for work to be done in parallel

Comprehensability
Can understand how a module works (in isolation)

Flexibility
Can be changed independently from each other

INFORMATION HIDING

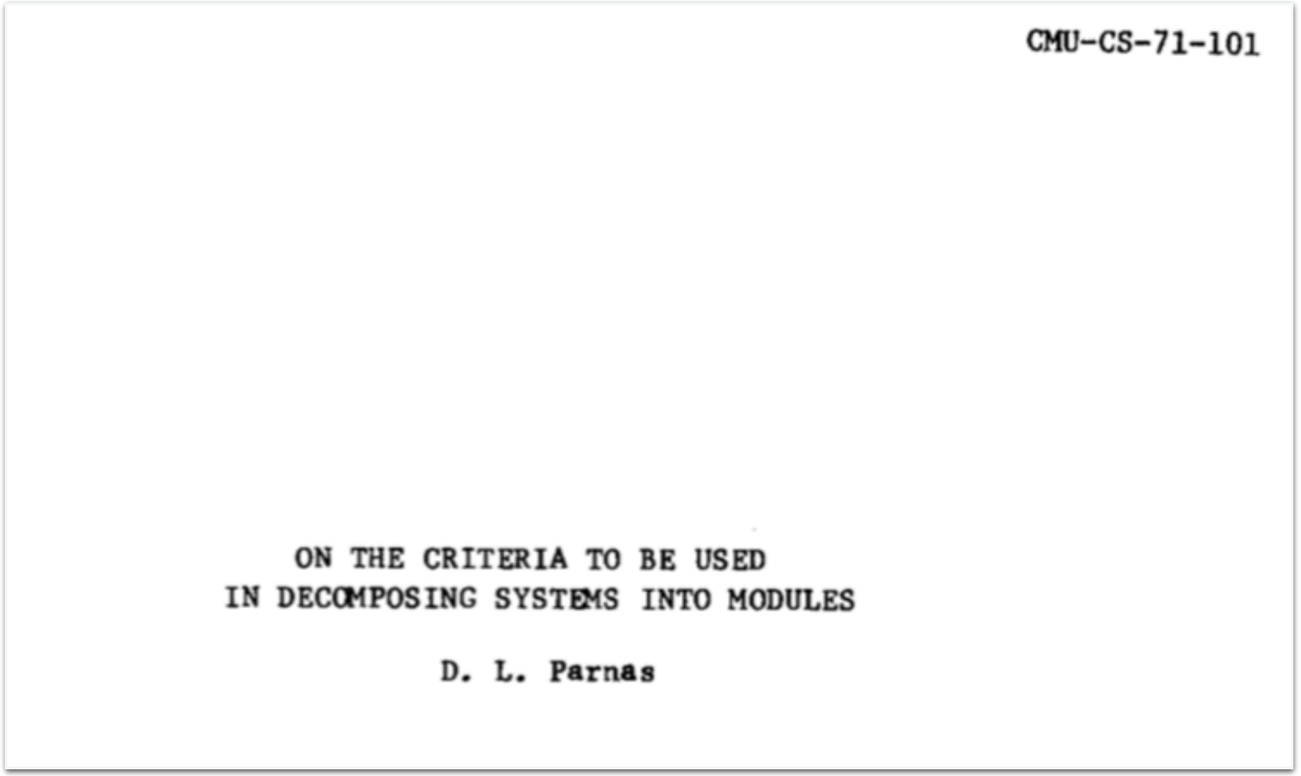
CMU-CS-71-101

ON THE CRITERIA TO BE USED
IN DECOMPOSING SYSTEMS INTO MODULES

D. L. Parnas

http://repository.cmu.edu/cgi/viewcontent.cgi?article=2979&context=compsci

INFORMATION HIDING



CMU-CS-71-101

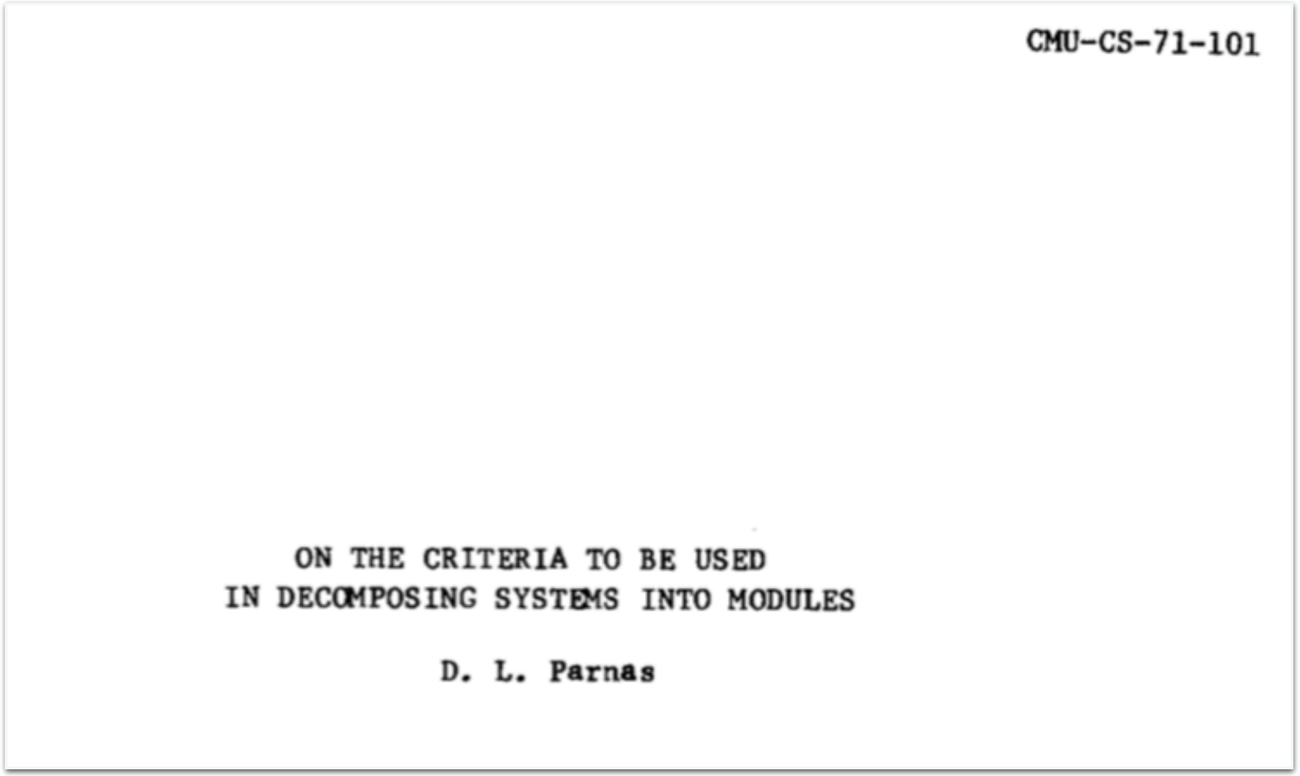
ON THE CRITERIA TO BE USED
IN DECOMPOSING SYSTEMS INTO MODULES

D. L. Parnas

Published in 1971

http://repository.cmu.edu/cgi/viewcontent.cgi?article=2979&context=compsci

INFORMATION HIDING


CMU-CS-71-101

ON THE CRITERIA TO BE USED
IN DECOMPOSING SYSTEMS INTO MODULES

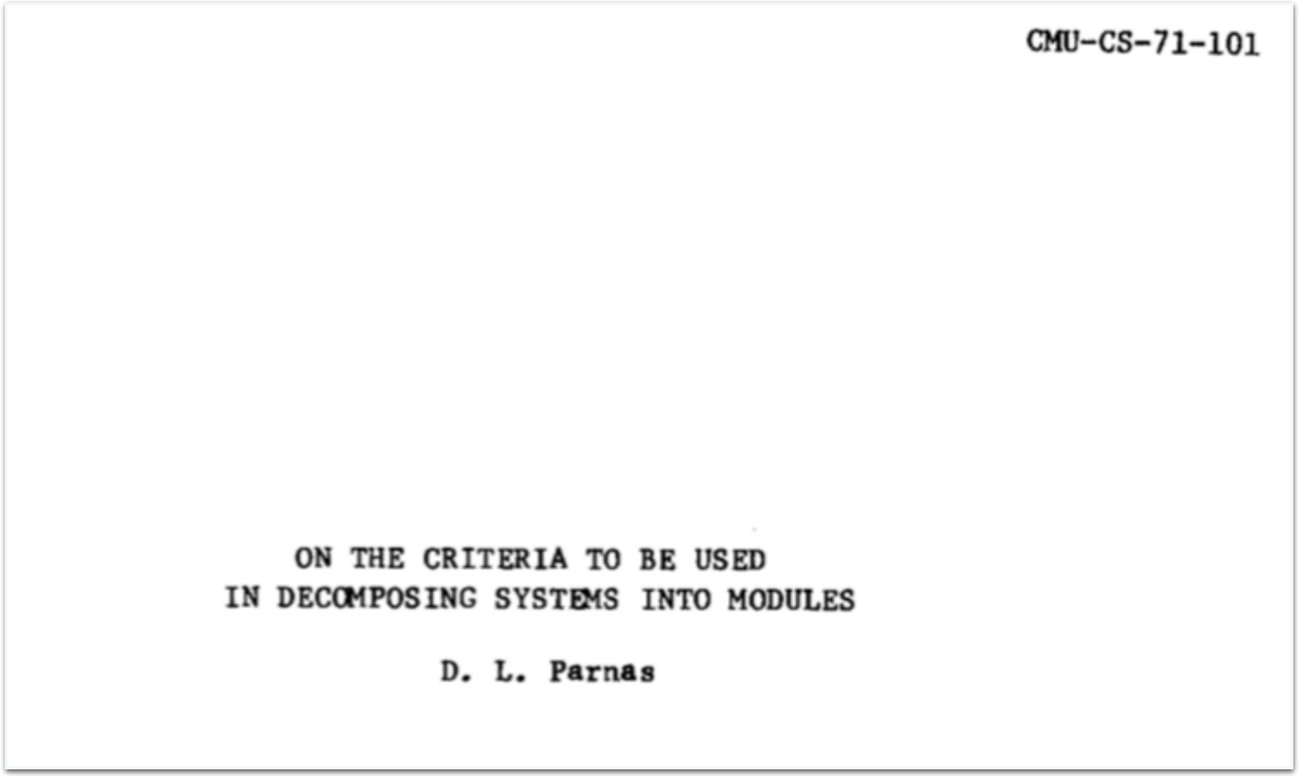
D. L. Parnas

Published in 1971

Looked at how best to define module boundaries

http://repository.cmu.edu/cgi/viewcontent.cgi?article=2979&context=compsci

INFORMATION HIDING


CMU-CS-71-101

ON THE CRITERIA TO BE USED
IN DECOMPOSING SYSTEMS INTO MODULES

D. L. Parnas

Published in 1971

Looked at how best to define module boundaries

Found that “information hiding” worked best

http://repository.cmu.edu/cgi/viewcontent.cgi?article=2979&context=compsci

WHAT IS INFORMATION HIDING?

WHAT IS INFORMATION HIDING?

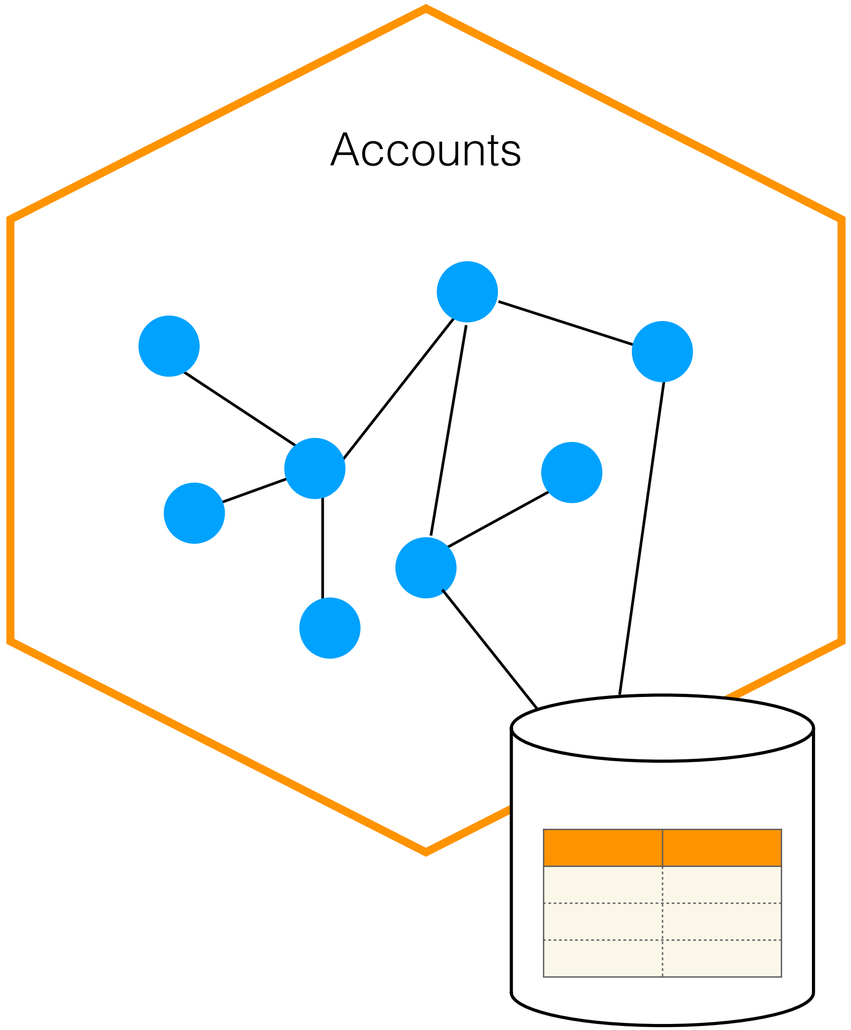
Hiding things which are most likely to change behind a stable boundary

WHAT IS INFORMATION HIDING?

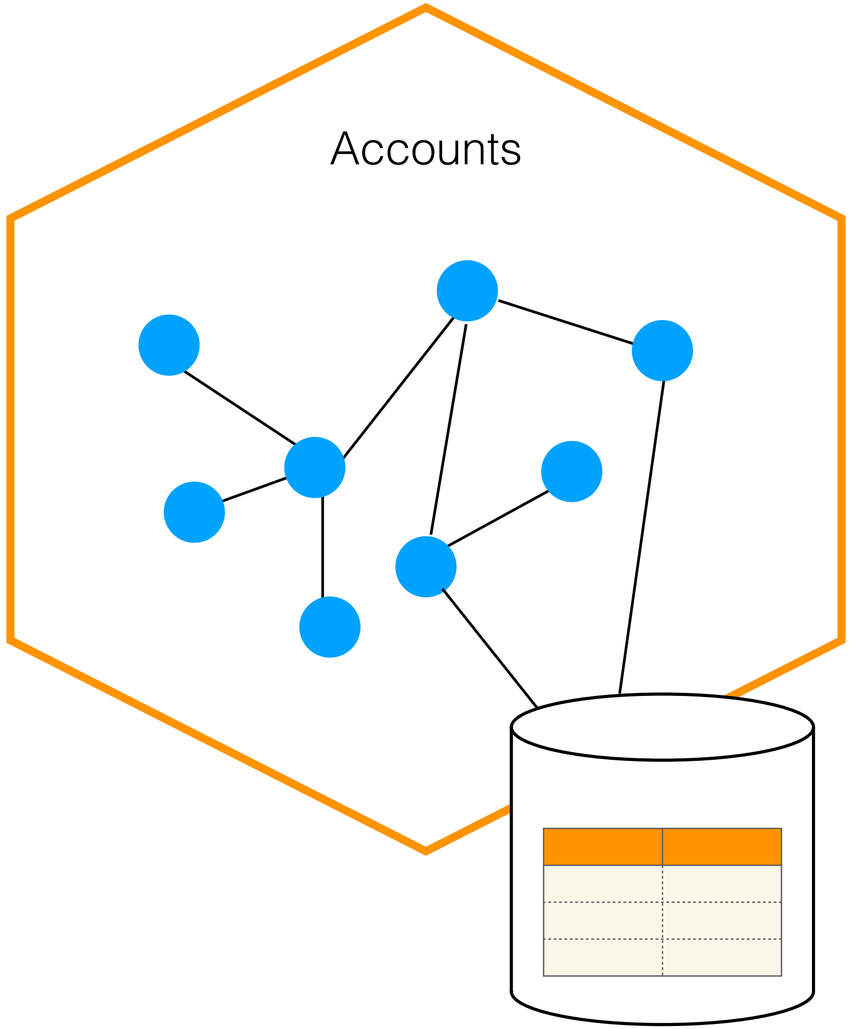
Hiding things which are most likely to change behind a stable boundary

Sam’s view: a default approach whereby everything is hidden unless explicitly needed

NOT HIDING?

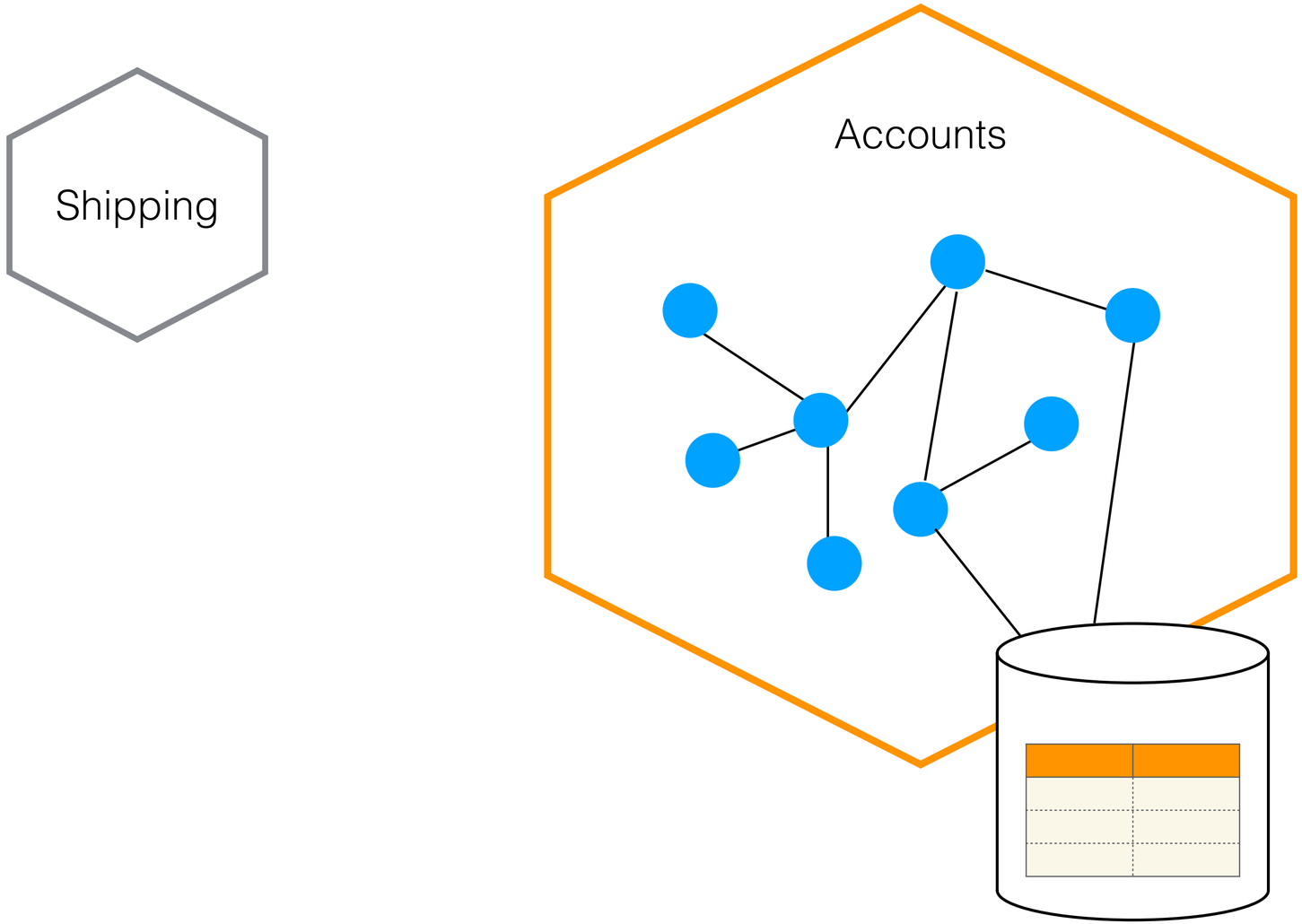


NOT HIDING?



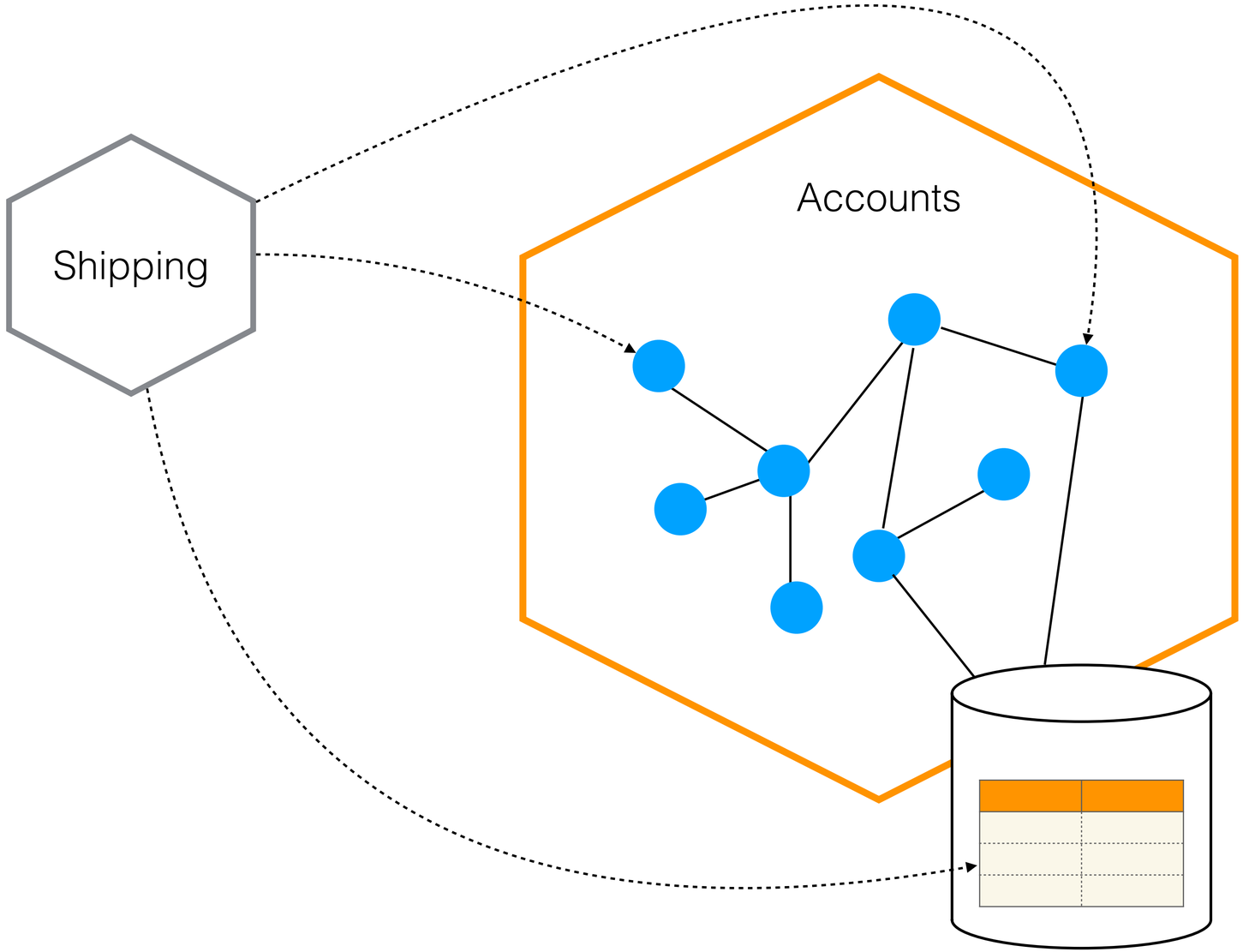
If an upstream consumer can reach into your internal implementation..

NOT HIDING?



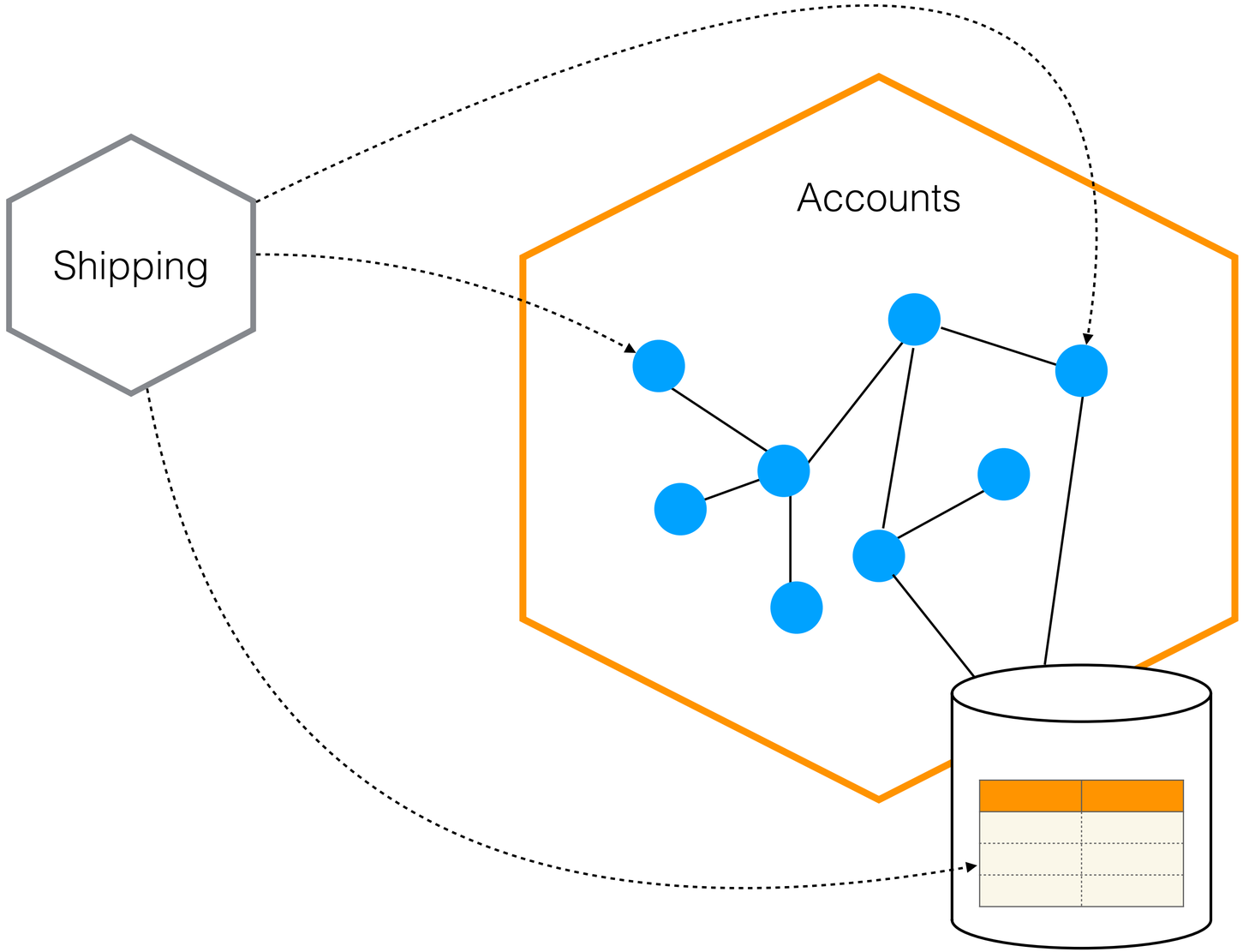
If an upstream consumer can reach into your internal implementation..

NOT HIDING?



If an upstream consumer can reach into your internal implementation..

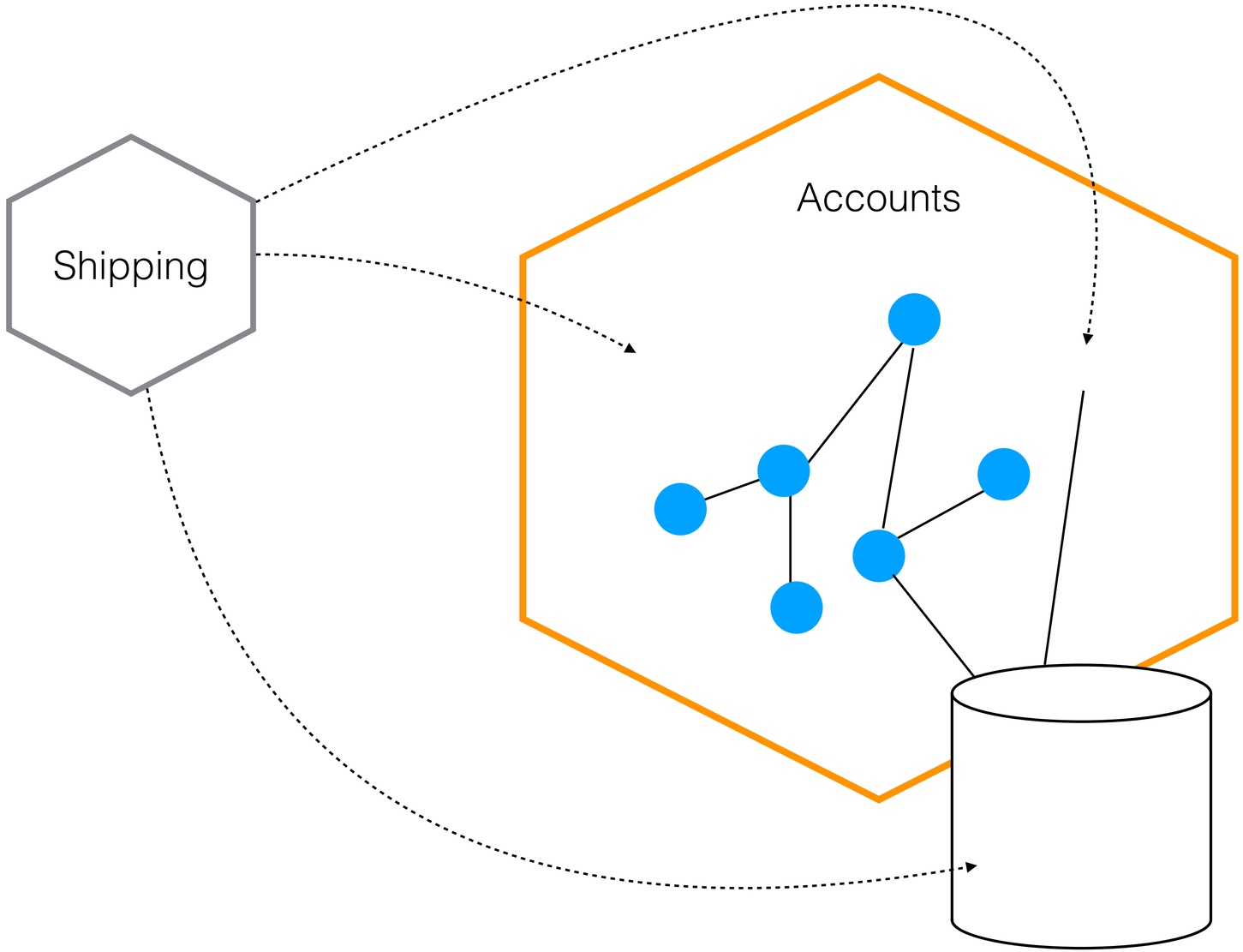
NOT HIDING?



If an upstream consumer can reach into your internal implementation..

…then you can't change this implementation without breaking the consumer

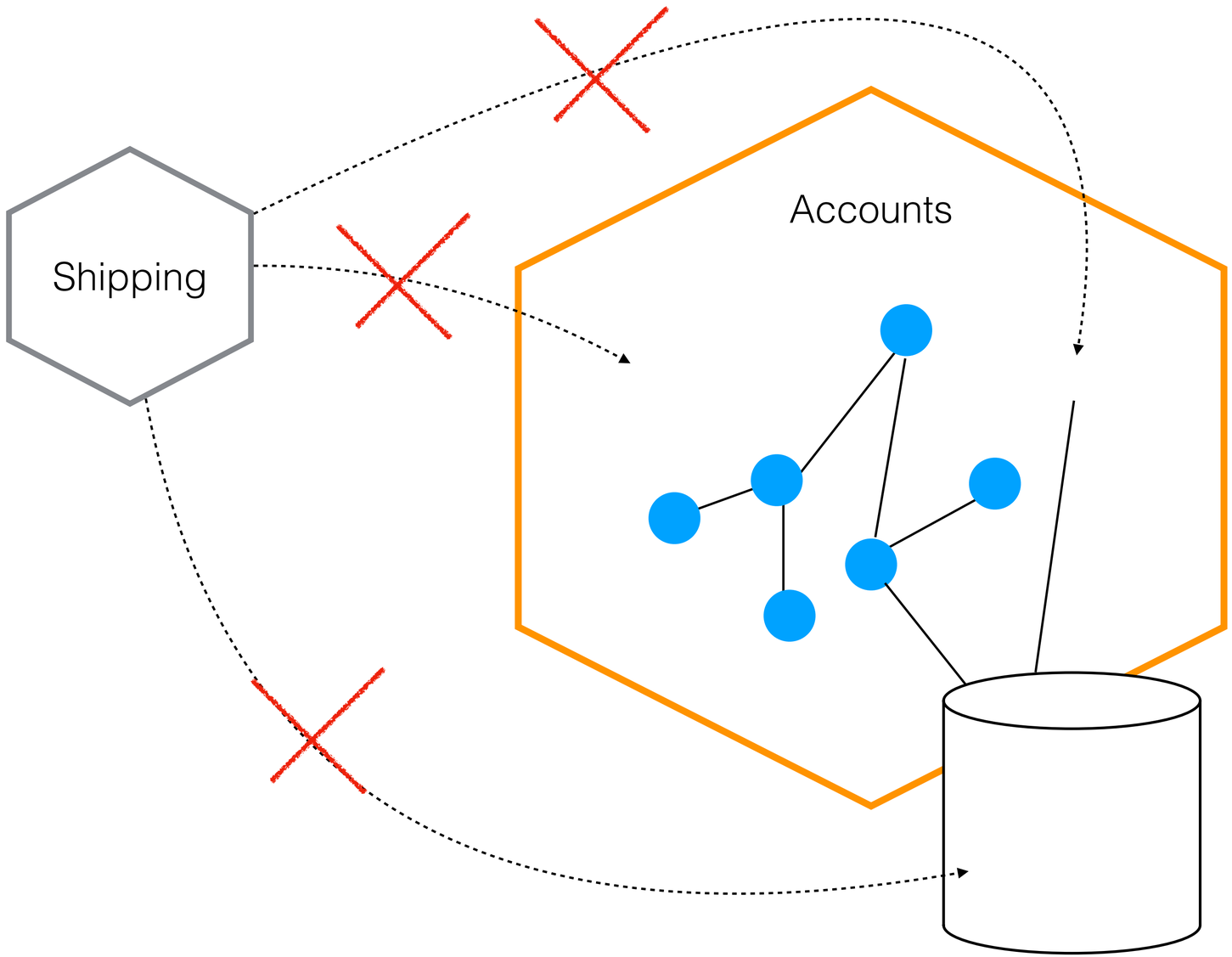
NOT HIDING?



If an upstream consumer can reach into your internal implementation..

…then you can't change this implementation without breaking the consumer

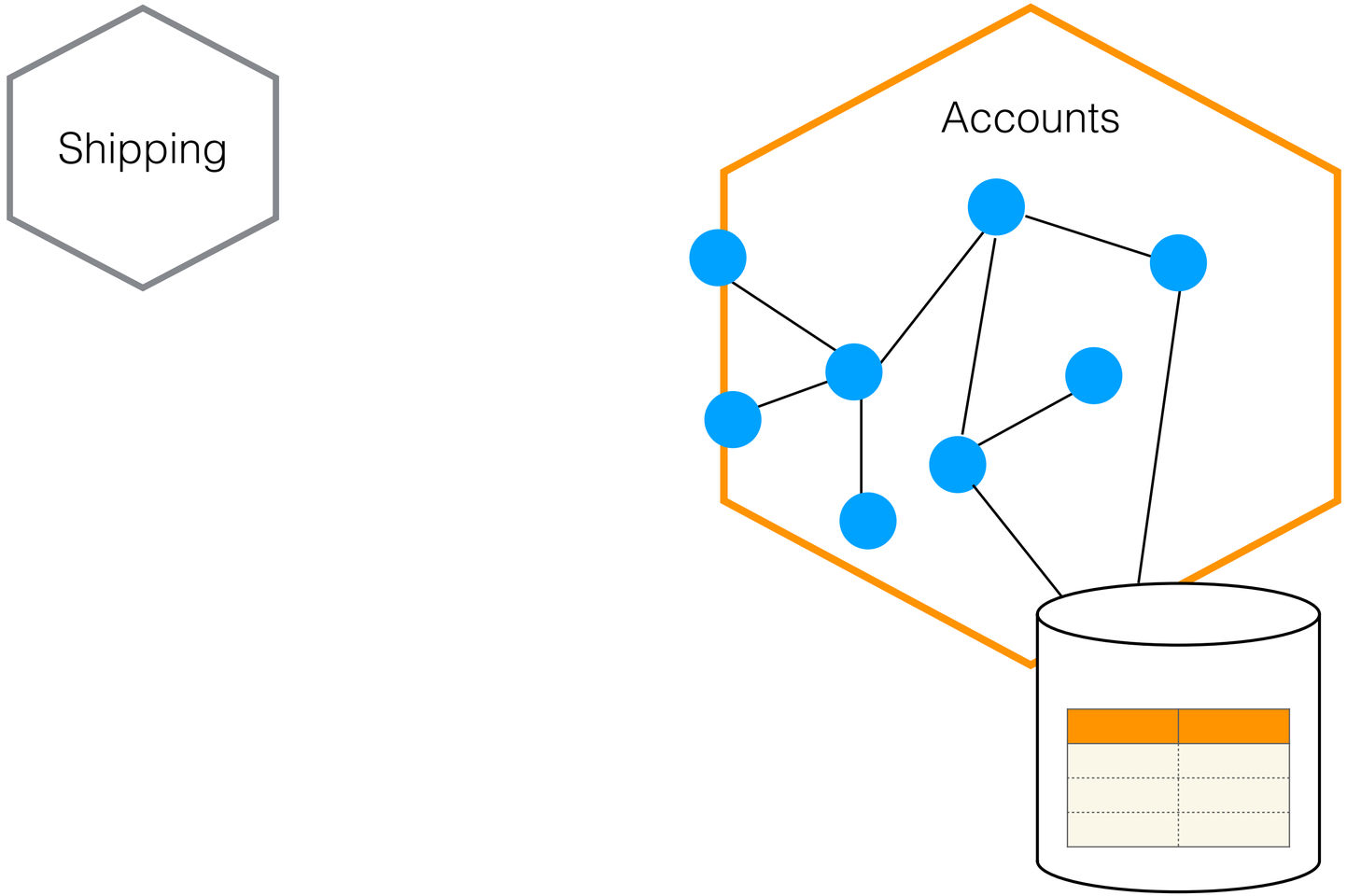
NOT HIDING?



If an upstream consumer can reach into your internal implementation..

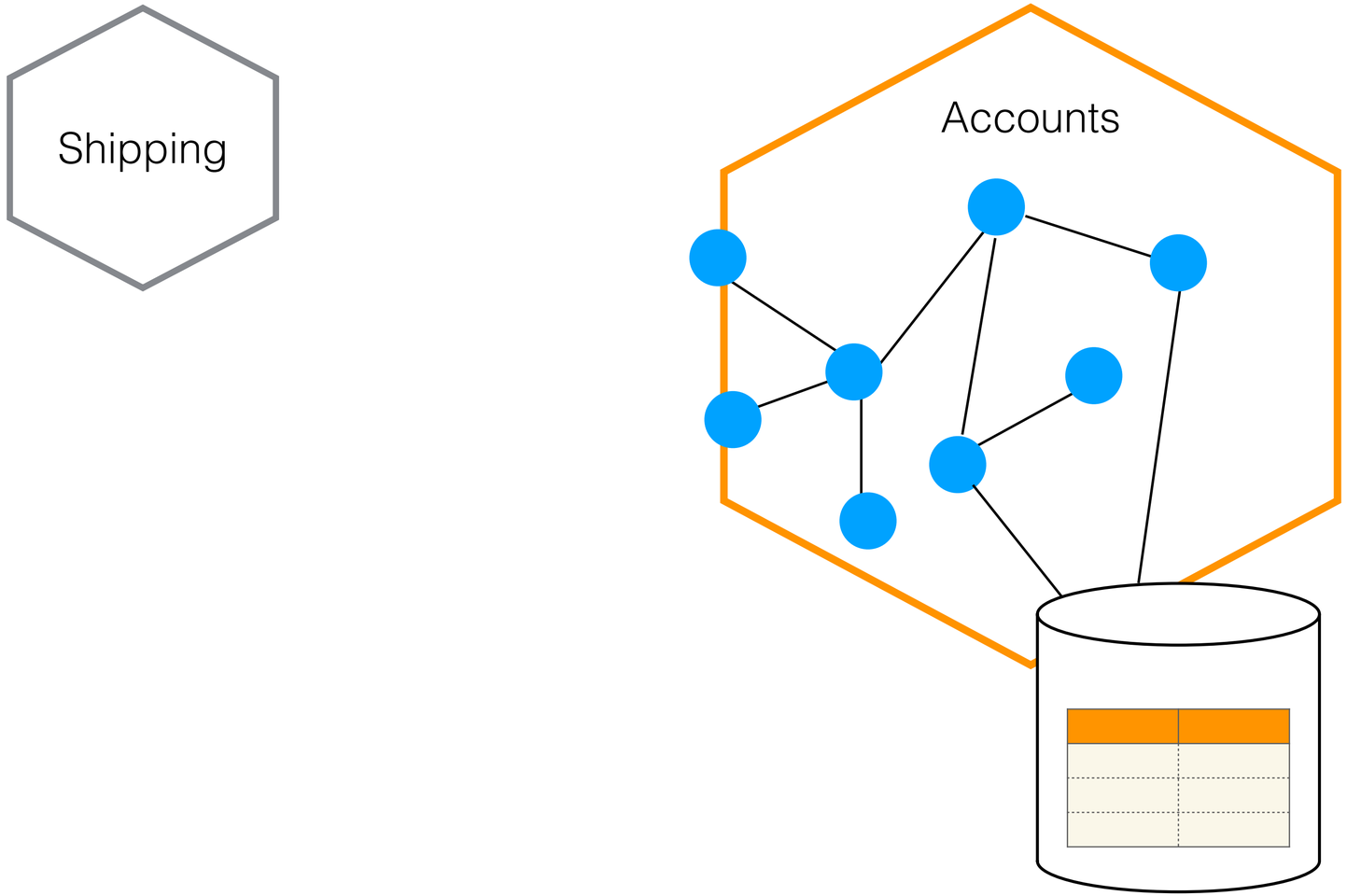
…then you can't change this implementation without breaking the consumer

HIDING!

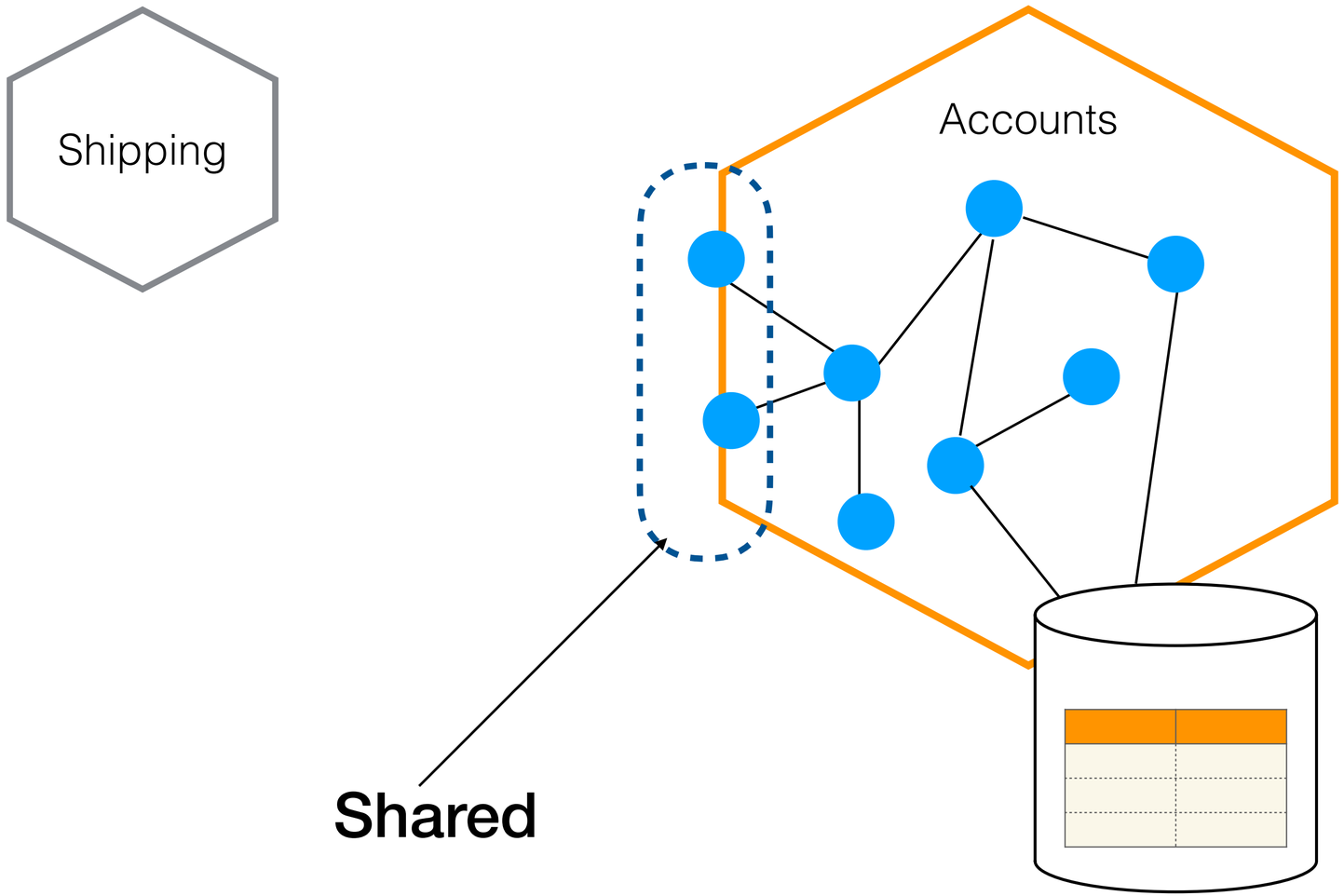


HIDING!

Hide your secrets!



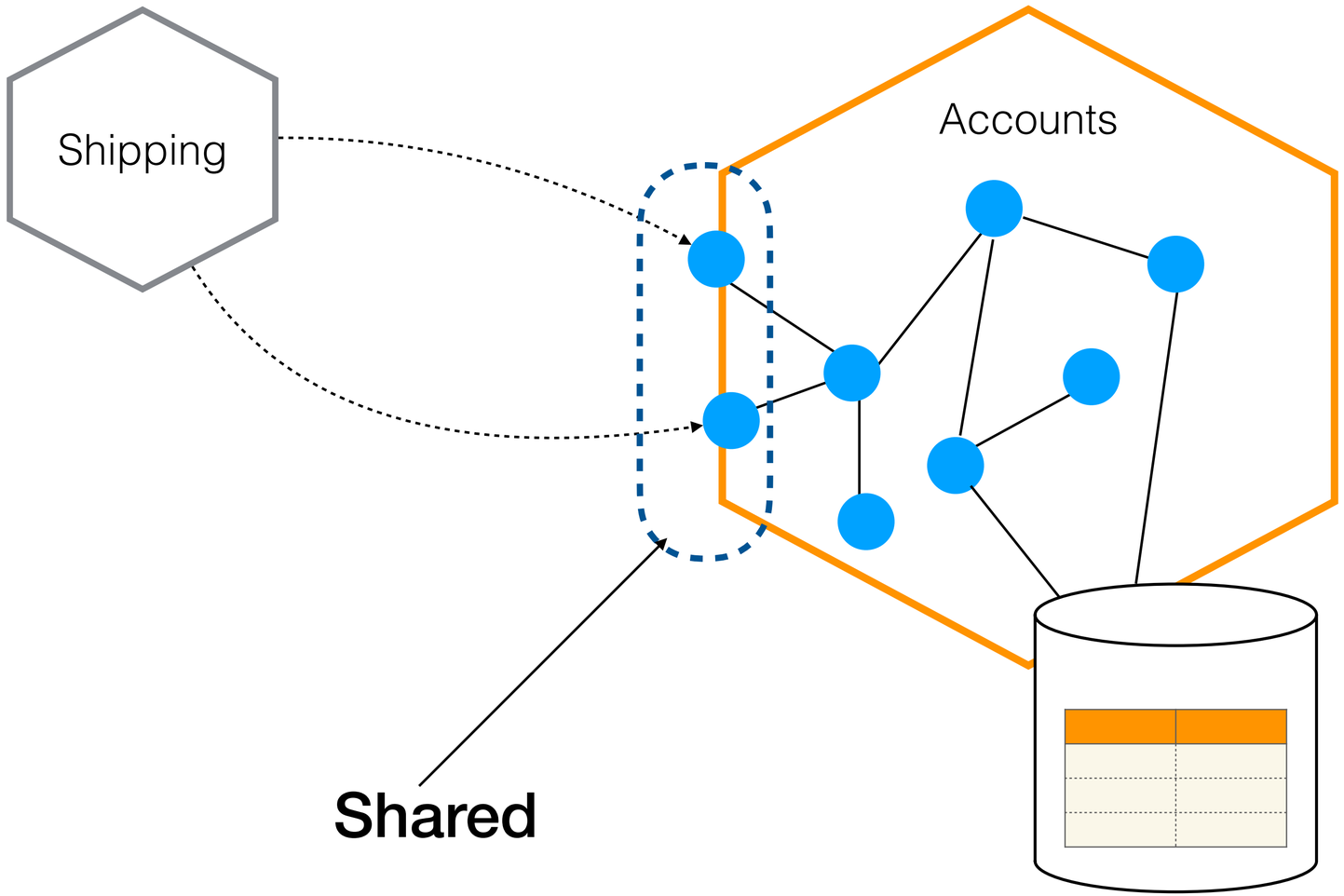
HIDING!



Hide your secrets!

Be explicit about what is shared, and what is hidden

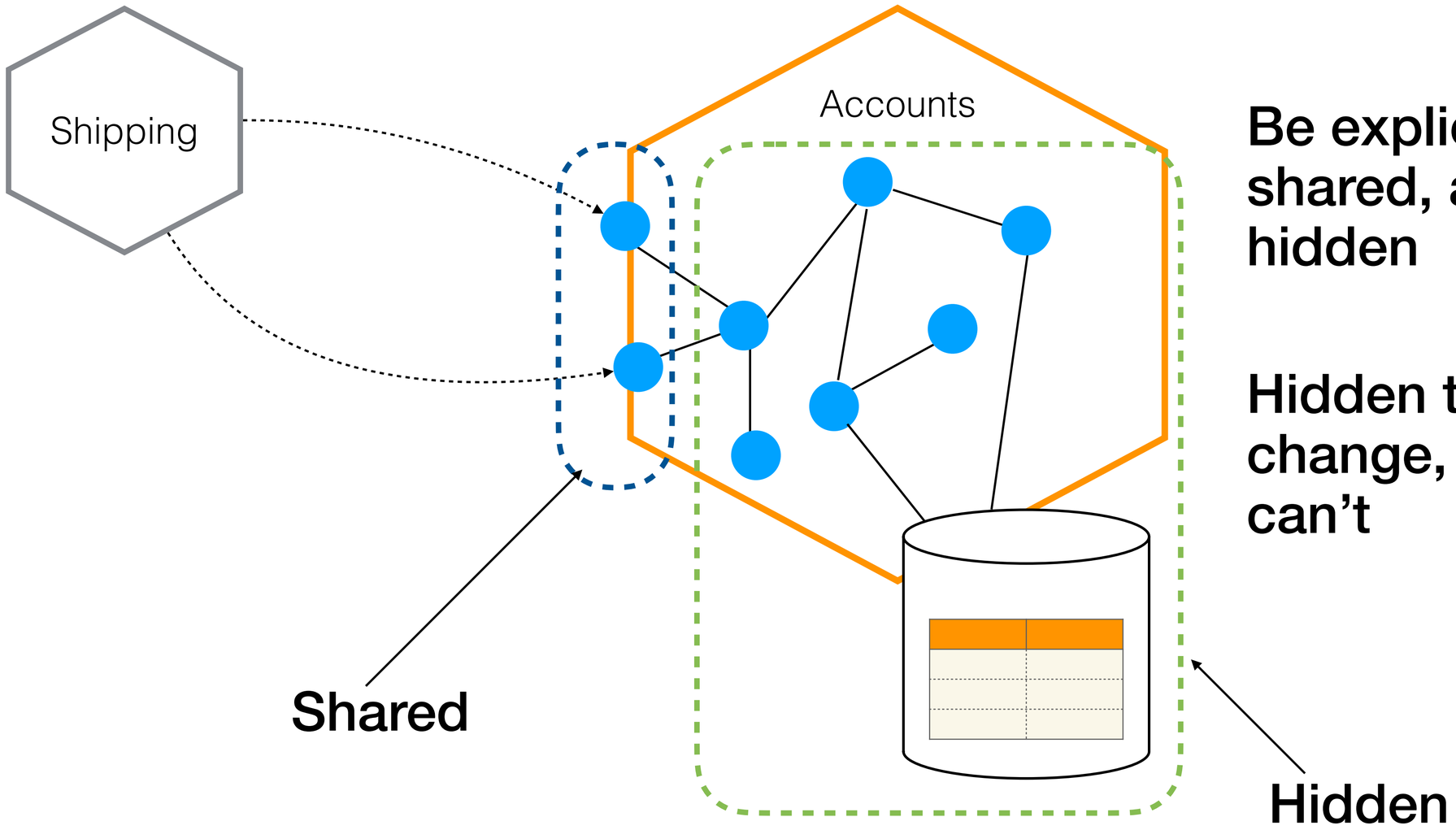
HIDING!



Hide your secrets!

Be explicit about what is shared, and what is hidden

HIDING!

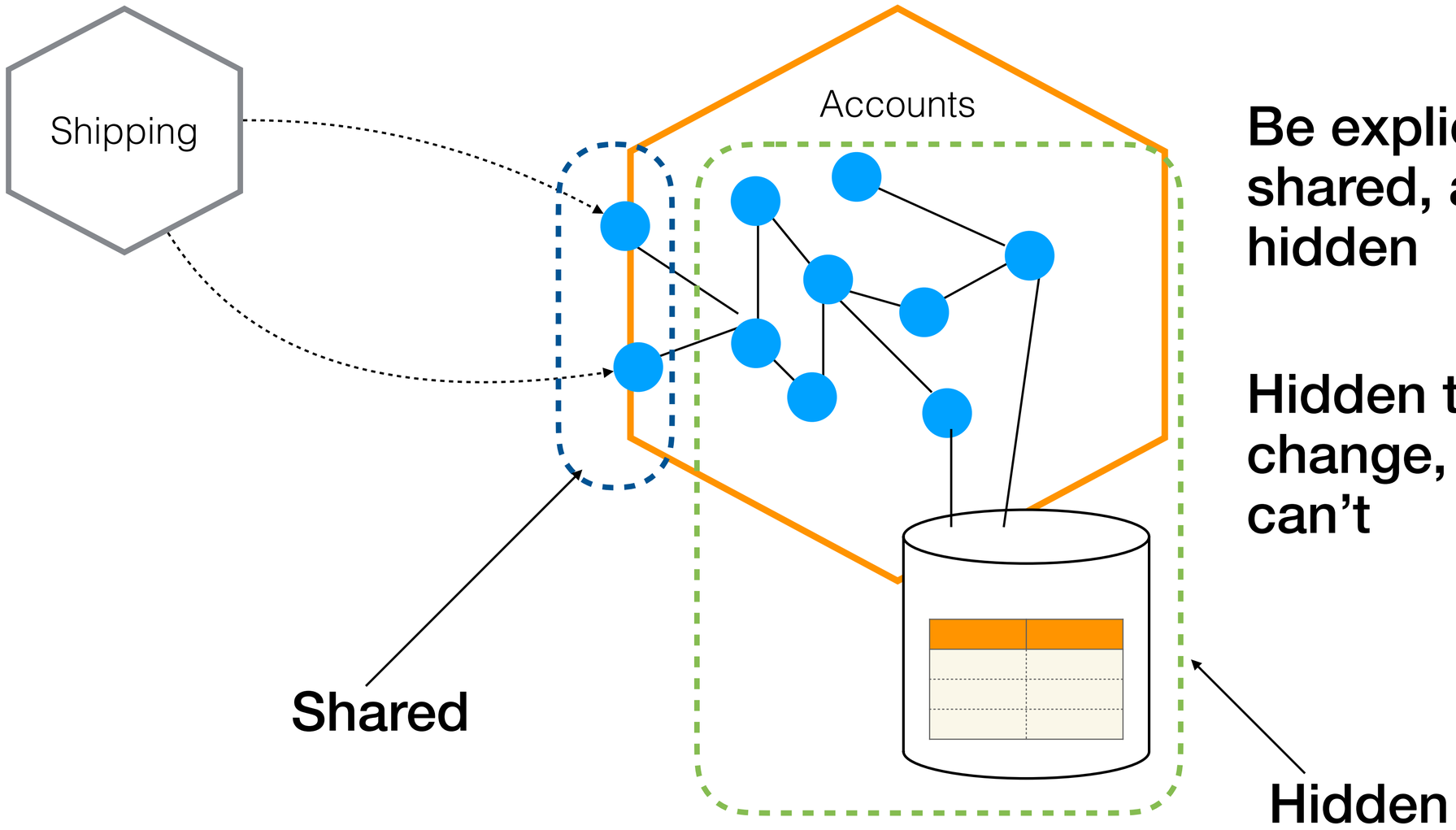


Hide your secrets!

Be explicit about what is shared, and what is hidden

Hidden things can change, shared things can't

HIDING!



Hide your secrets!

Be explicit about what is shared, and what is hidden

Hidden things can change, shared things can't

AND CONNECTIONS BETWEEN THEM?

INFORMATION DISTRIBUTION ASPECTS OF DESIGN METHODOLOGY *

D.L. PARNAS

Computer Science Department, Carnegie-Mellon University, Pittsburgh, Pennsylvania, USA

The role of documentation in the design and implementation of complex systems is explored, resulting in suggestions in sharp contrast with current practice. The concept of system structure is studied by examining the meaning of the phrase "connections between modules". It is shown that several system design goals (each suggesting a partial time ordering of the decisions) may be inconsistent. Some properties of programmers are discussed. System documentation which makes all information accessible to anyone working on the project, is discussed. The thesis that such information "broadcasting" is harmful, that it is helpful if most system information can be hidden from most programmers, is supported by use of the above mentioned considerations as well as by examples.

1. INTRODUCTION

Papers on design methodology assume (1) that the methods used in system design strongly affect the quality of the final product; and (2) by selecting an appropriate methodology we can avoid many of the problems previously encountered in constructing large systems.

Under the heading "Design Methodology" a number of separate topics can be distinguished:

1. The order in which design decisions are made [1, 2, 3, 6].
2. The characteristics of the final product (e.g., what constitutes "good structure" for a system) [4, 5, 6, 7].
3. Methods of detecting errors in design decisions shortly after they are made [1, 2, 3, 5, 8, 9].
4. Specification techniques [12, 13].

2. STRUCTURE DEFINED

The word "structure" is used to refer to a partial description of a system. A structure description shows the system divided into a set of *modules*, gives some characteristics of each module, and specifies some *connections* between the modules. Any given system admits many such descriptions. Since structure descriptions are not unique, our usage of "module" does not allow a precise definition parallel to that of "subroutine" in software or "card" in hardware. The definitions of those words delineate a class of object, but not the definition of "module". Nevertheless, "module" is useful in the same manner that "unit" is in military or economic discussions. We shall continue to use "module" without a precise defintion. It refers to portions of a system indicated in a description of that system. Its precise definition is not

@samnewman

AND CONNECTIONS BETWEEN THEM?

INFORMATION DISTRIBUTION ASPECTS OF DESIGN METHODOLOGY *

D.L. PARNAS

Computer Science Department, Carnegie-Mellon University, Pittsburgh, Pennsylvania, USA

The role of documentation in the design and implementation of complex systems is explored, resulting in suggestions in sharp contrast with current practice. The concept of system structure is studied by examining the meaning of the phrase "connections between modules". It is shown that several system design goals (each suggesting a partial time ordering of the decisions) may be inconsistent. Some properties of programmers are discussed. System documentation which makes all information accessible to anyone working on the project, is discussed. The thesis that such information "broadcasting" is harmful, that it is helpful if most system information can be hidden from most programmers, is supported by use of the above mentioned considerations as well as by examples.

1. INTRODUCTION

Papers on design methodology assume (1) that the methods used in system design strongly affect the quality of the final product; and (2) by selecting an appropriate methodology we can avoid many of the problems previously encountered in constructing large systems.

Under the heading "Design Methodology" a number of separate topics can be distinguished:

1. The order in which design decisions are made [1, 2, 3, 6].
2. The characteristics of the final product (e.g., what constitutes "good structure" for a system) [4, 5, 6, 7].
3. Methods of detecting errors in design decisions shortly after they are made [1, 2, 3, 5, 8, 9].
4. Specification techniques [12, 13].

2. STRUCTURE DEFINED

The word "structure" is used to refer to a partial description of a system. A structure description shows the system divided into a set of *modules*, gives some characteristics of each module, and specifies some *connections* between the modules. Any given system admits many such descriptions. Since structure descriptions are not unique, our usage of "module" does not allow a precise definition parallel to that of "subroutine" in software or "card" in hardware. The definitions of those words delineate a class of object, but not the definition of "module". Nevertheless, "module" is useful in the same manner that "unit" is in military or economic discussions. We shall continue to use "module" without a precise defintion. It refers to portions of a system indicated in a description of that system. Its precise definition is not

"The connections between modules are the assumptions which the modules make about each other."

- David Parnas

@samnewman

AND CONNECTIONS BETWEEN THEM?

INFORMATION DISTRIBUTION ASPECTS OF DESIGN METHODOLOGY *

D.L. PARNAS

Computer Science Department, Carnegie-Mellon University, Pittsburgh, Pennsylvania, USA

The role of documentation in the design and implementation of complex systems is explored, resulting in suggestions in sharp contrast with current practice. The concept of system structure is studied by examining the meaning of the phrase "connections between modules". It is shown that several system design goals (each suggesting a partial time ordering of the decisions) may be inconsistent. Some properties of programmers are discussed. System documentation which makes all information accessible to anyone working on the project, is discussed. The thesis that such information "broadcasting" is harmful, that it is helpful if most system information can be hidden from most programmers, is supported by use of the above mentioned considerations as well as by examples.

1. INTRODUCTION

Papers on design methodology assume (1) that the methods used in system design strongly affect the quality of the final product; and (2) by selecting an appropriate methodology we can avoid many of the problems previously encountered in constructing large systems.

Under the heading "Design Methodology" a number of separate topics can be distinguished:

1. The order in which design decisions are made [1, 2, 3, 6].
2. The characteristics of the final product (e.g., what constitutes "good structure" for a system) [4, 5, 6, 7].
3. Methods of detecting errors in design decisions shortly after they are made [1, 2, 3, 5, 8, 9].
4. Specification techniques [12, 13].

2. STRUCTURE DEFINED

The word "structure" is used to refer to a partial description of a system. A structure description shows the system divided into a set of *modules*, gives some characteristics of each module, and specifies some *connections* between the modules. Any given system admits many such descriptions. Since structure descriptions are not unique, our usage of "module" does not allow a precise definition parallel to that of "subroutine" in software or "card" in hardware. The definitions of those words delineate a class of object, but not the definition of "module". Nevertheless, "module" is useful in the same manner that "unit" is in military or economic discussions. We shall continue to use "module" without a precise defintion. It refers to portions of a system indicated in a description of that system. Its precise definition is not

"The connections between modules are the assumptions which the modules make about each other."

- David Parnas

Information hiding is about reducing the assumptions one module has for another

@samnewman

The more you hide, the easier it is to maintain backwards compatibility

ACCIDENTAL BREAKAGE

Code

```
public class Customer {
  private int id;
  private String name;
  private int age;
  …
}
```

ACCIDENTAL BREAKAGE

Code

```
public class Customer {
  private int id;
  private String name;
  private int age;
  …
}
```

ACCIDENTAL BREAKAGE

Code

```
public class Customer {
  private int id;
  private String name;
  private int age;
  …
}
```

JSON

```
{
  “id” : 123,
  “name” : “sam”,
  “age” : 15
}
```

ACCIDENTAL BREAKAGE

Code

```
public class Customer {
  private int id;
  private String name;
  private int age;
  …
}
```

JSON

```
{
  “id” : 123,
  “name” : “sam”,
  “age” : 15
}
```

ACCIDENTAL BREAKAGE

Code

```
public class Customer {
  private int id;
  private String name;
  private int age;
  …
}
```

JSON

```
{
  “id” : 123,
  “name” : “sam”,
  “age” : 15
}
```

ACCIDENTAL BREAKAGE

Code

```
public class Customer {
  private int id;
  private String name;
  private int age;
  …
}
```

JSON

```
{
  “id” : 123,
  “name” : “sam”,
  “age” : 15
}
```

Code

```
public class Customer {
  private int id;
  private String name;
  private LocalDate dob;
  …
}
```

ACCIDENTAL BREAKAGE

Code

```
public class Customer {
  private int id;
  private String name;
  private int age;
  …
}
```

JSON

```
{
  “id” : 123,
  “name” : “sam”,
  “age” : 15
}
```

Code

```
public class Customer {
  private int id;
  private String name;
  private LocalDate dob;
  …
}
```

ACCIDENTAL BREAKAGE

Code

```
public class Customer {
  private int id;
  private String name;
  private int age;
  …
}
```

JSON

```
{
  “id” : 123,
  “name” : “sam”,
  “age” : 15
}
```

Code

```
public class Customer {
  private int id;
  private String name;
  private LocalDate dob;
  …
}
```

JSON

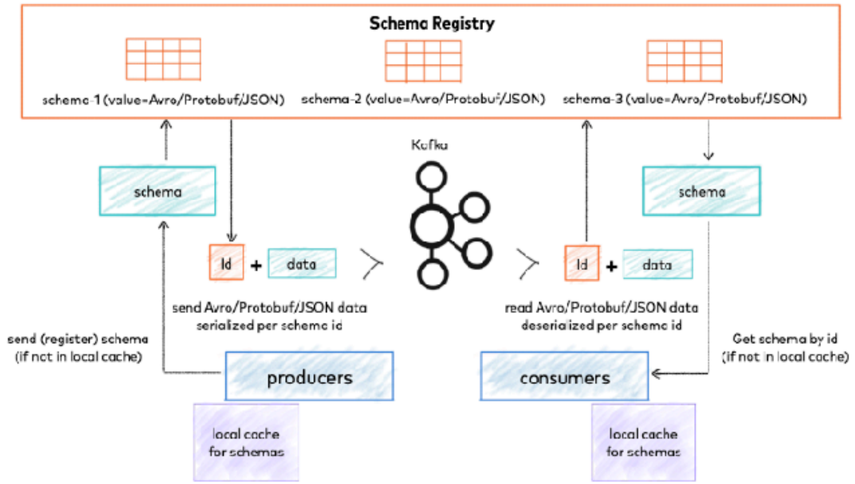
```
{
  “id” : 123,
  “name” : “sam”,
  “dob” : 15/02/1974
}
```

SCHEMA CHECKING

Schema Management

Confluent Schema Registry provides a serving layer for your metadata. It provides a RESTful interface for storing and retrieving your Avro®, JSON Schema, and Protobuf schemas. It stores a versioned history of all schemas based on a specified subject name strategy, provides multiple compatibility settings and allows evolution of schemas according to the configured compatibility settings and expanded support for these schema types. It provides serializers that plug into Apache Kafka® clients that handle schema storage and retrieval for Kafka messages that are sent in any of the supported formats.

Schema Registry lives outside of and separately from your Kafka brokers. Your producers and consumers still talk to Kafka to publish and read data (messages) to topics. Concurrently, they can also talk to Schema Registry to send and retrieve schemas that describe the data models for the messages.



Confluent Schema Registry for storing and retrieving schemas

https://docs.confluent.io/current/schema-registry/index.html

README.md

OpenAPI-diff

Compare two OpenAPI specifications (3.x) and render the difference to HTML plaintext, or Markdown files.

Main Build passing | quality gate passed | maven-central v2.0.0-beta.6 | Slack Join the chat room

docker build automated | docker build passing | version v2.0.0-beta.6

Requirements

- Java 8

Feature

- Supports OpenAPI spec v3.0.
- Depth comparison of parameters, responses, endpoint, http method (GET,POST,PUT,DELETE...)
- Supports swagger api Authorization
- Render difference of property with Expression Language
- HTML & Markdown render

Maven

Available on Maven Central

```
<dependency>
  <groupId>org.openapitools.openapidiff</groupId>
  <artifactId>openapi-diff-core</artifactId>
  <version>${openapi-diff-version}</version>
</dependency>
```

https://github.com/OpenAPITools/openapi-diff

POLL: HOW MANY OF YOU SHARE THE SAME DATABASE BETWEEN MICROSERVICES?

NA (we don’t have microservices)

No - we don’t do this at all

We share databases between microservices - and it’s fine!

We share databases between microservices, and it causes us pain!

3/4 Reasons to use microservices

POLL: DO YOU KNOW WHY YOU ARE USING MICROSERVICES?

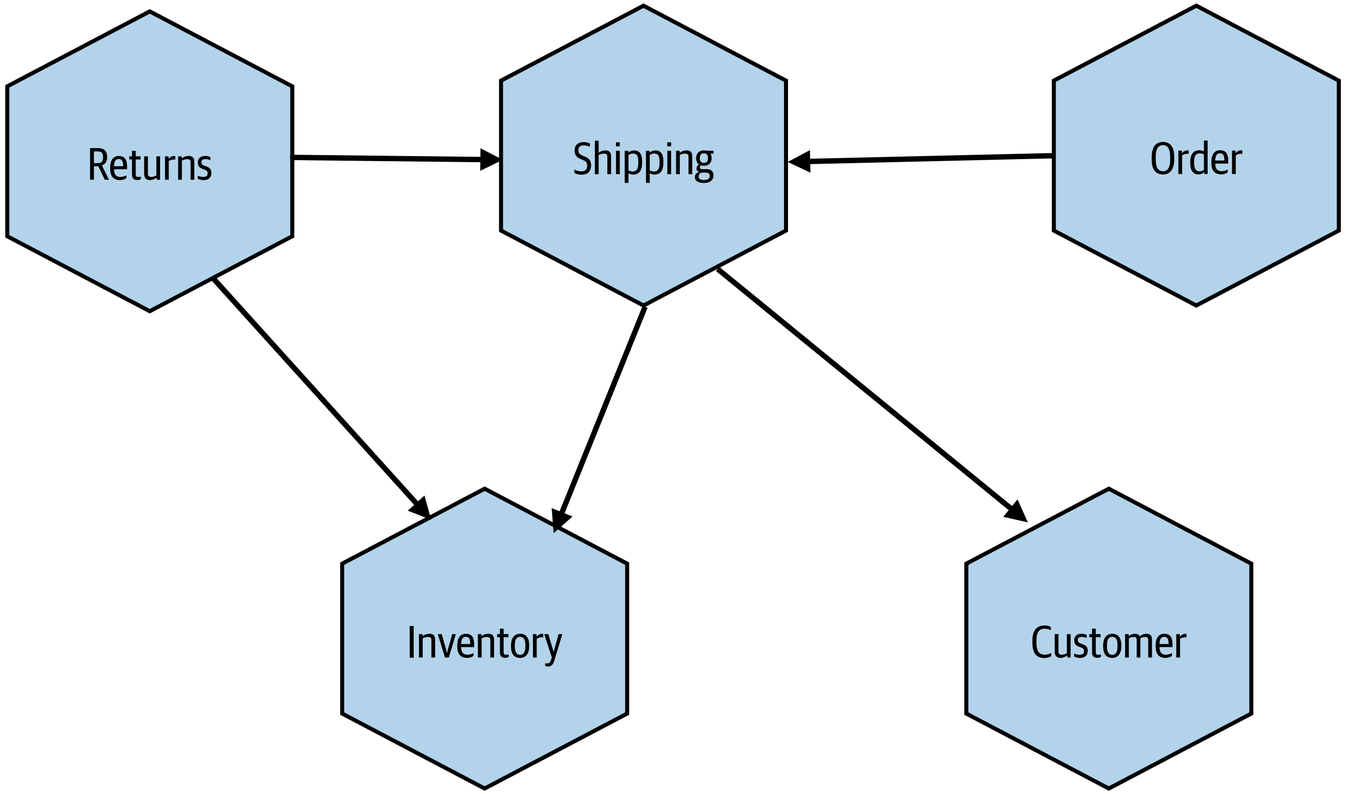
NA (we don’t have microservices)

No idea

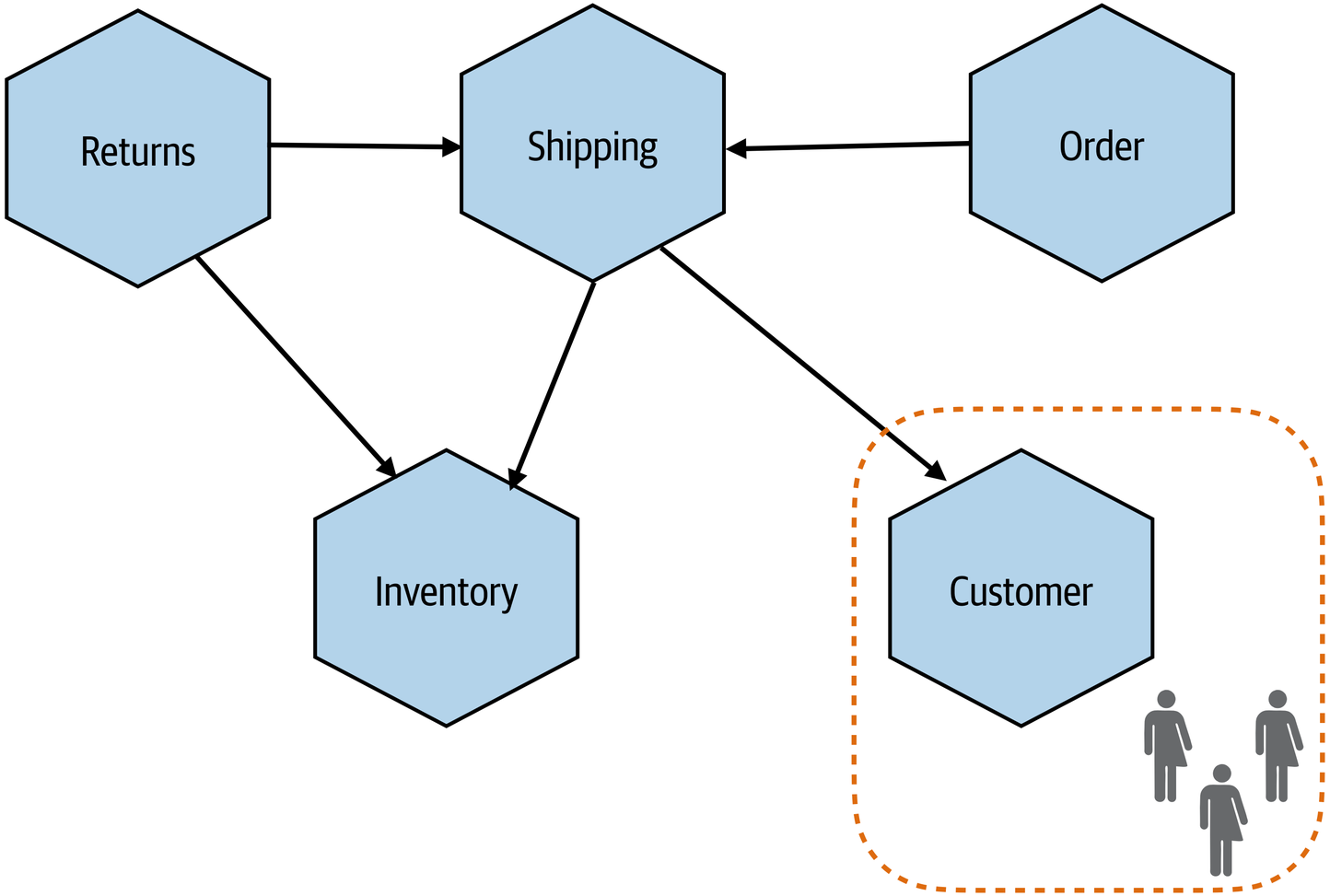
I have a vague idea

Yes, it’s super clear to me!

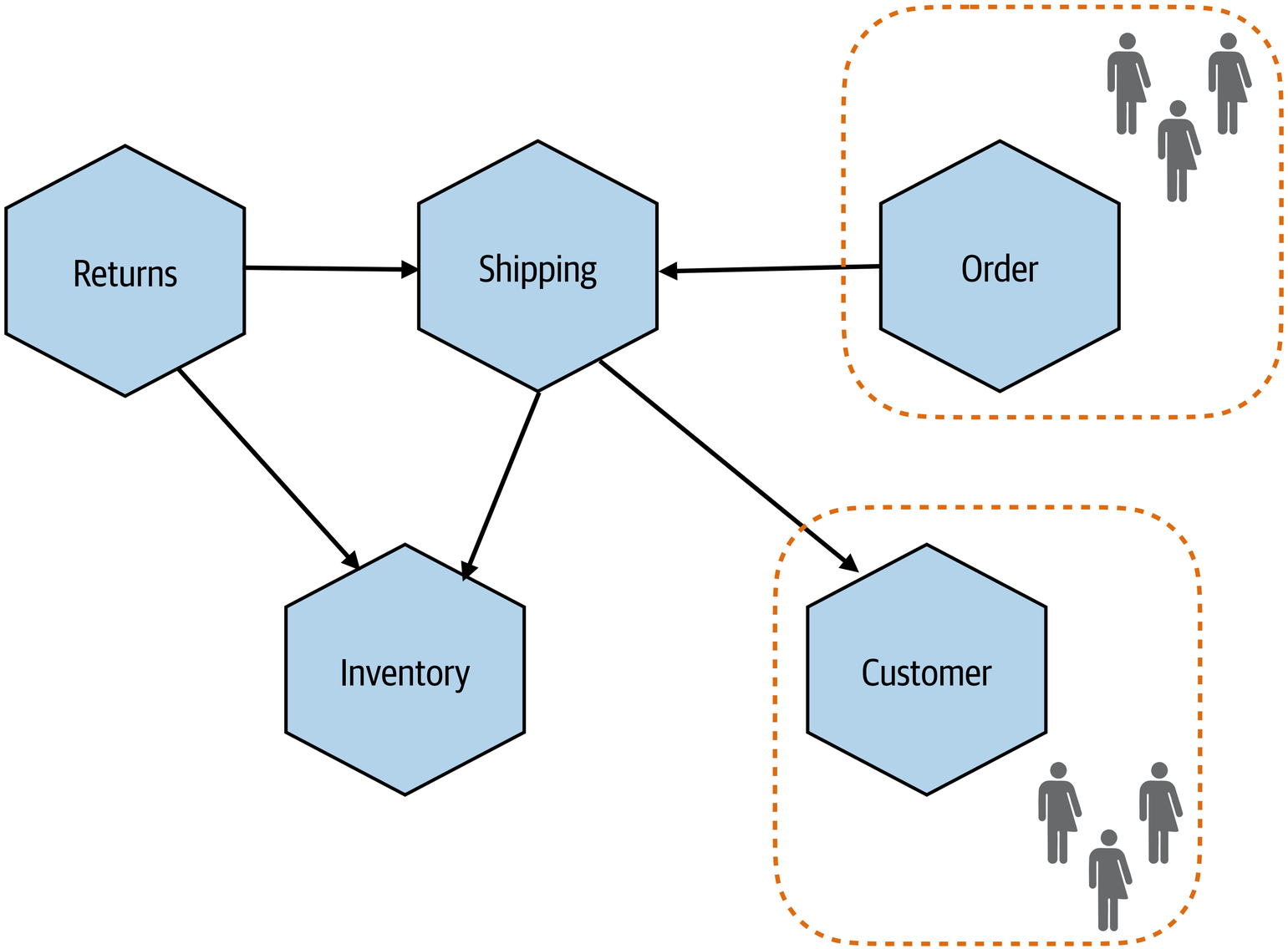
INDEPENDENT DEVELOPMENT & DELIVERY



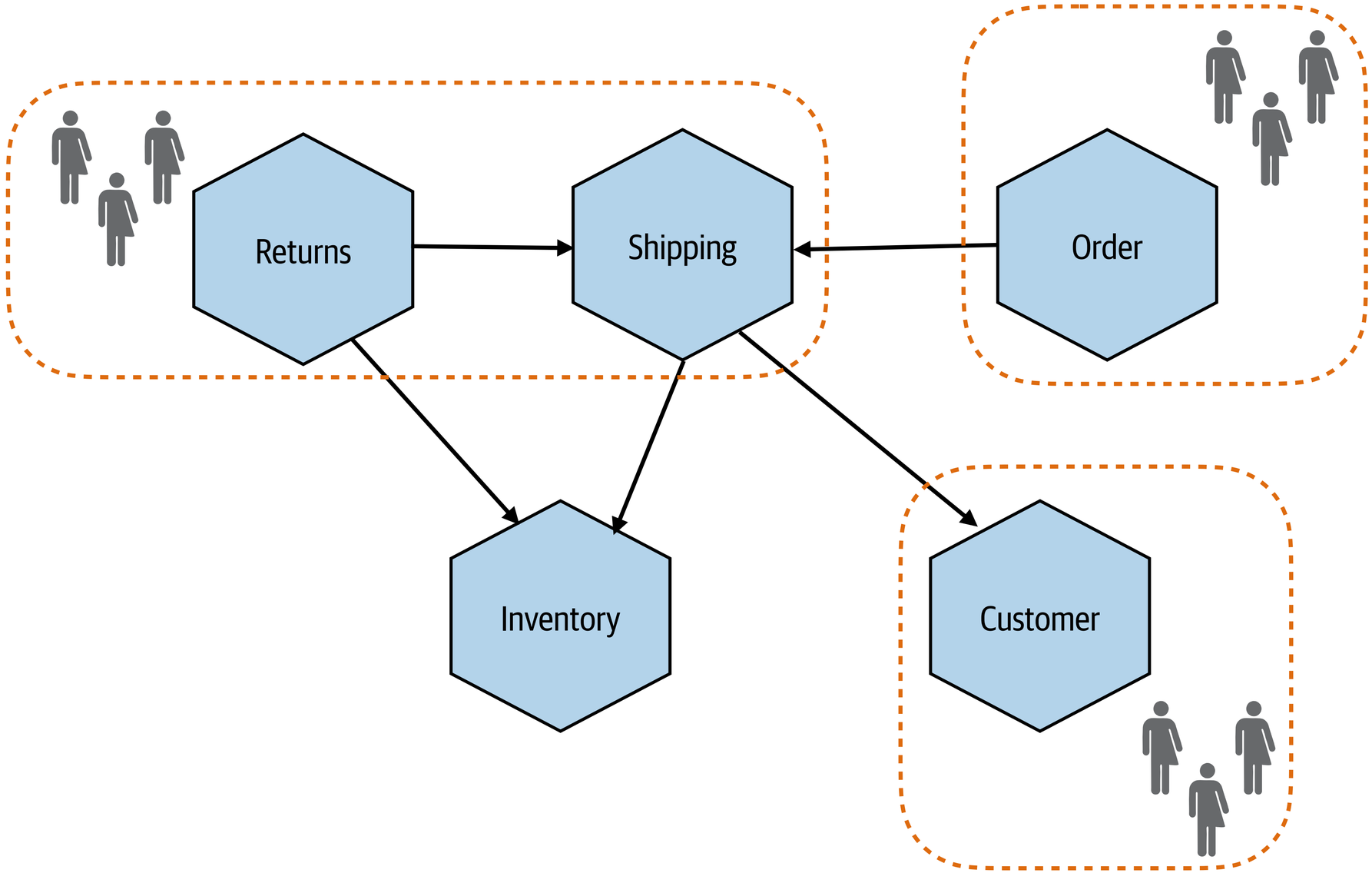
INDEPENDENT DEVELOPMENT & DELIVERY



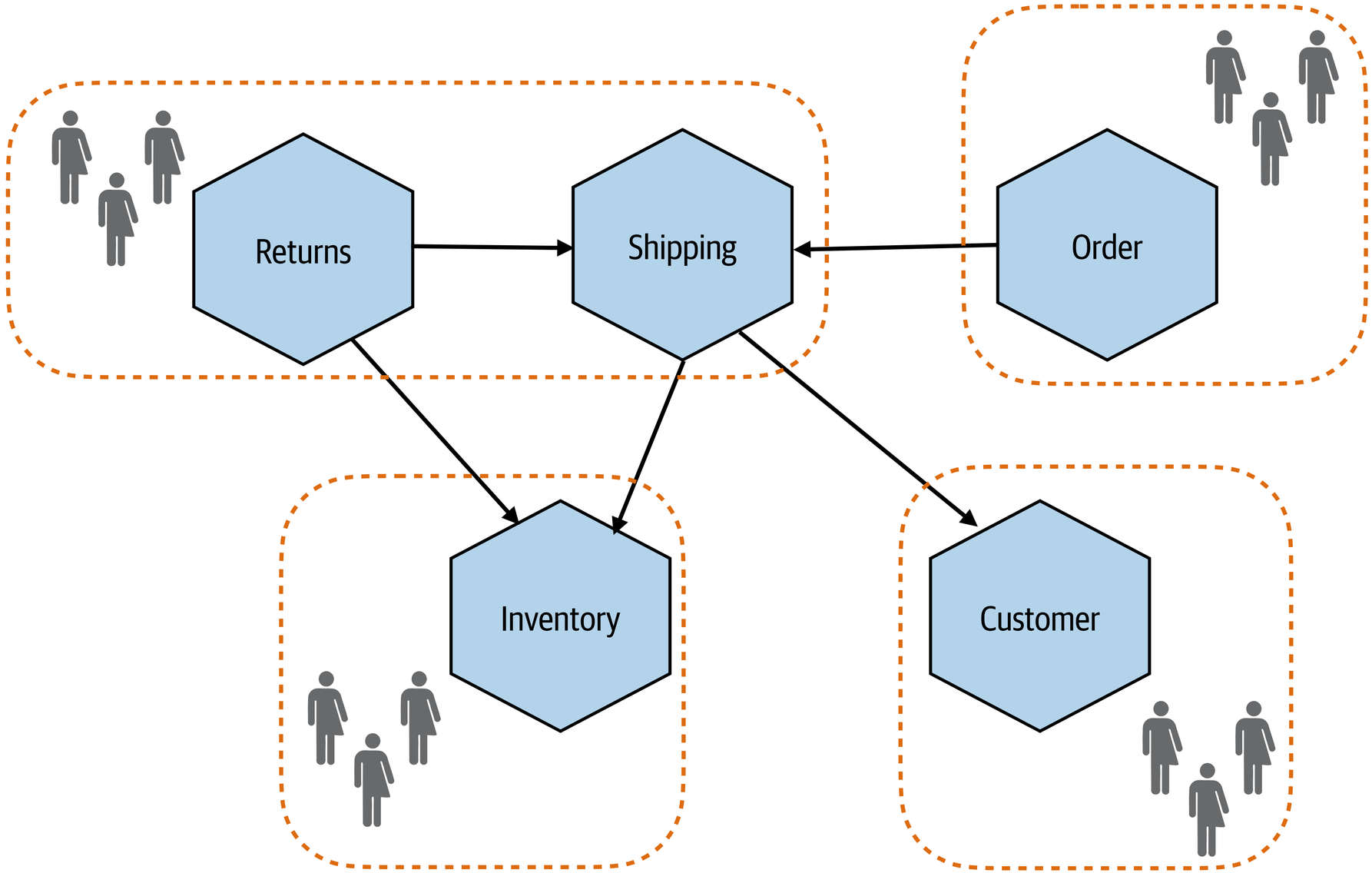
INDEPENDENT DEVELOPMENT & DELIVERY



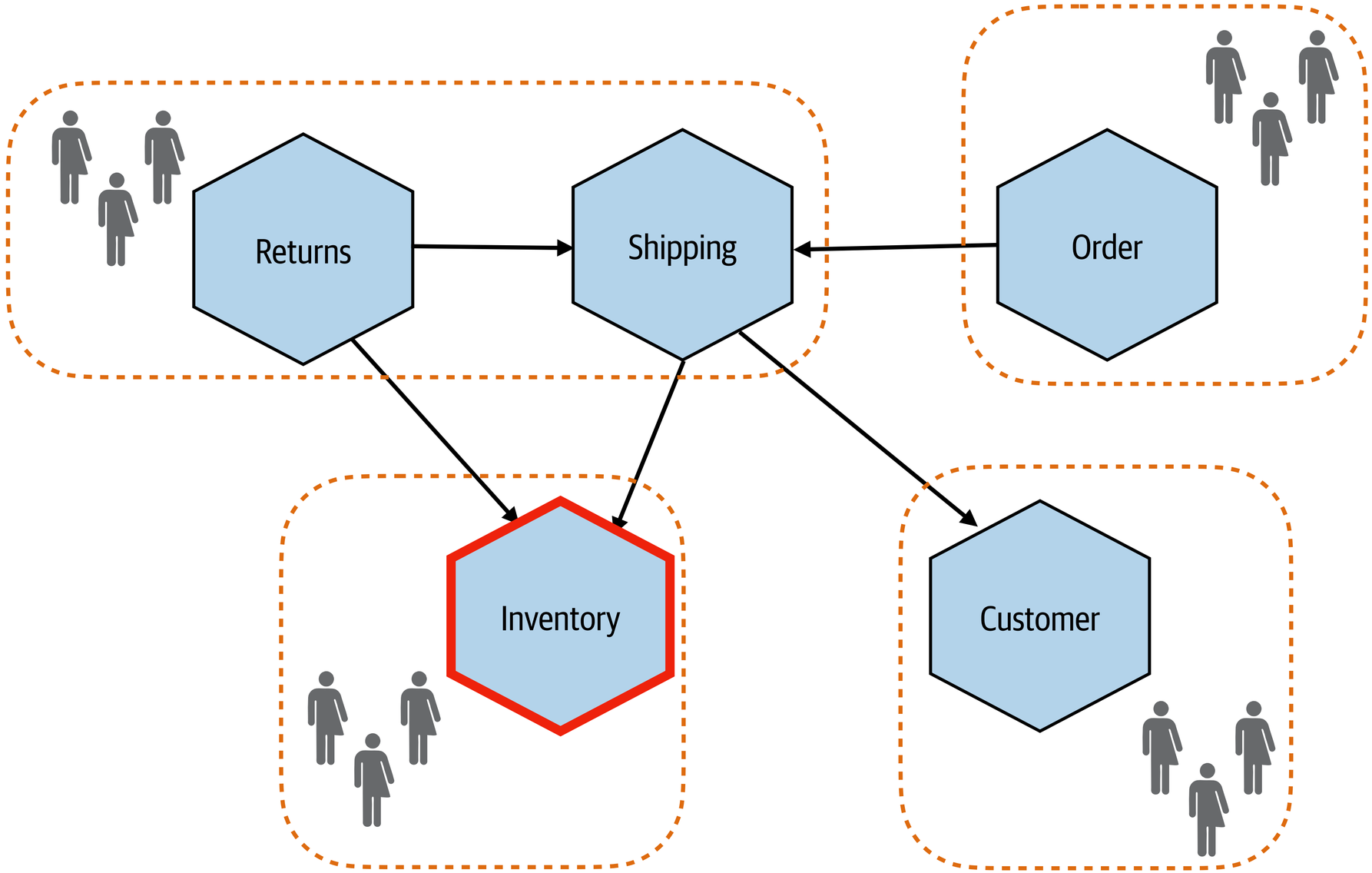
INDEPENDENT DEVELOPMENT & DELIVERY



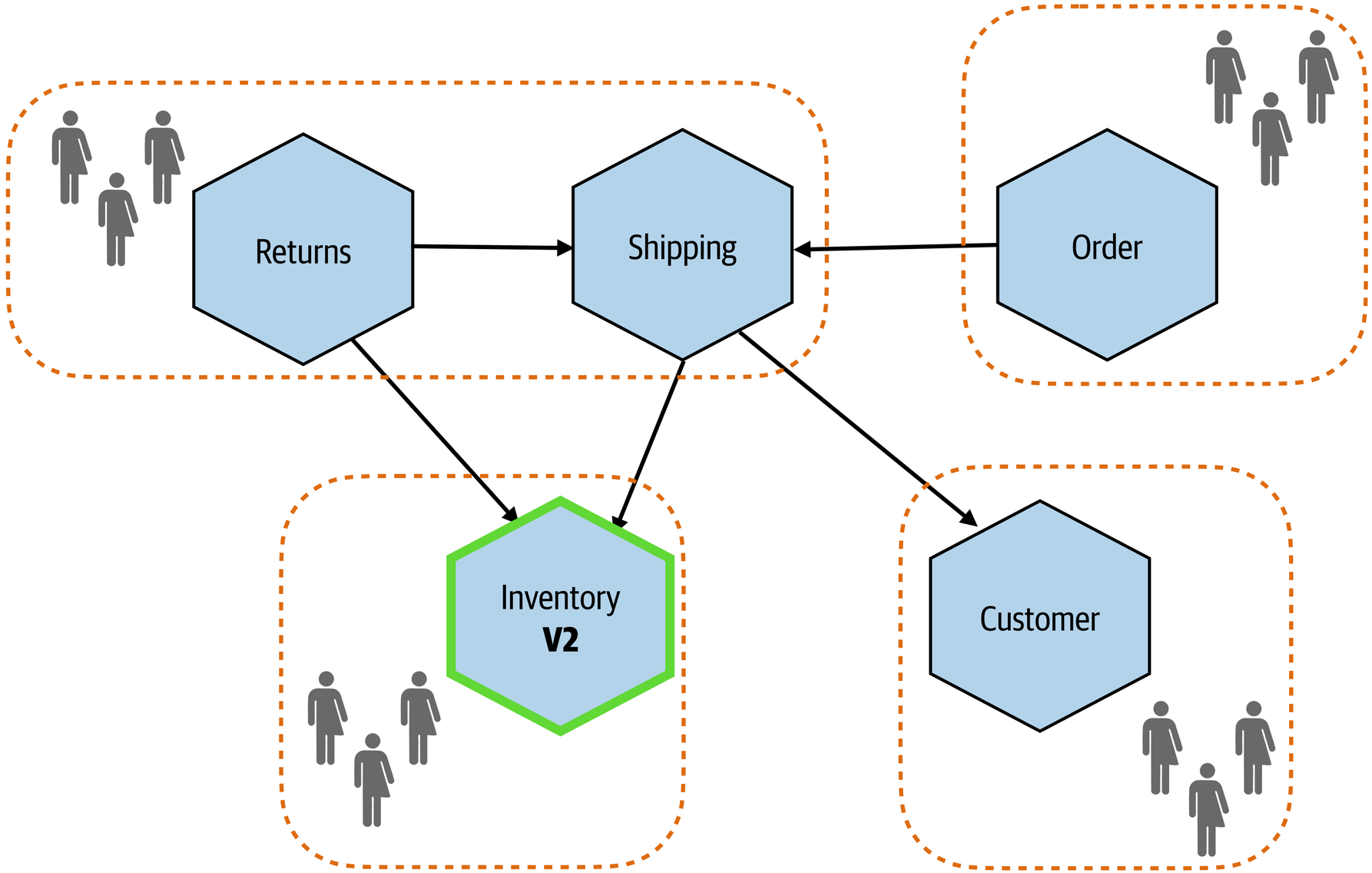
INDEPENDENT DEVELOPMENT & DELIVERY



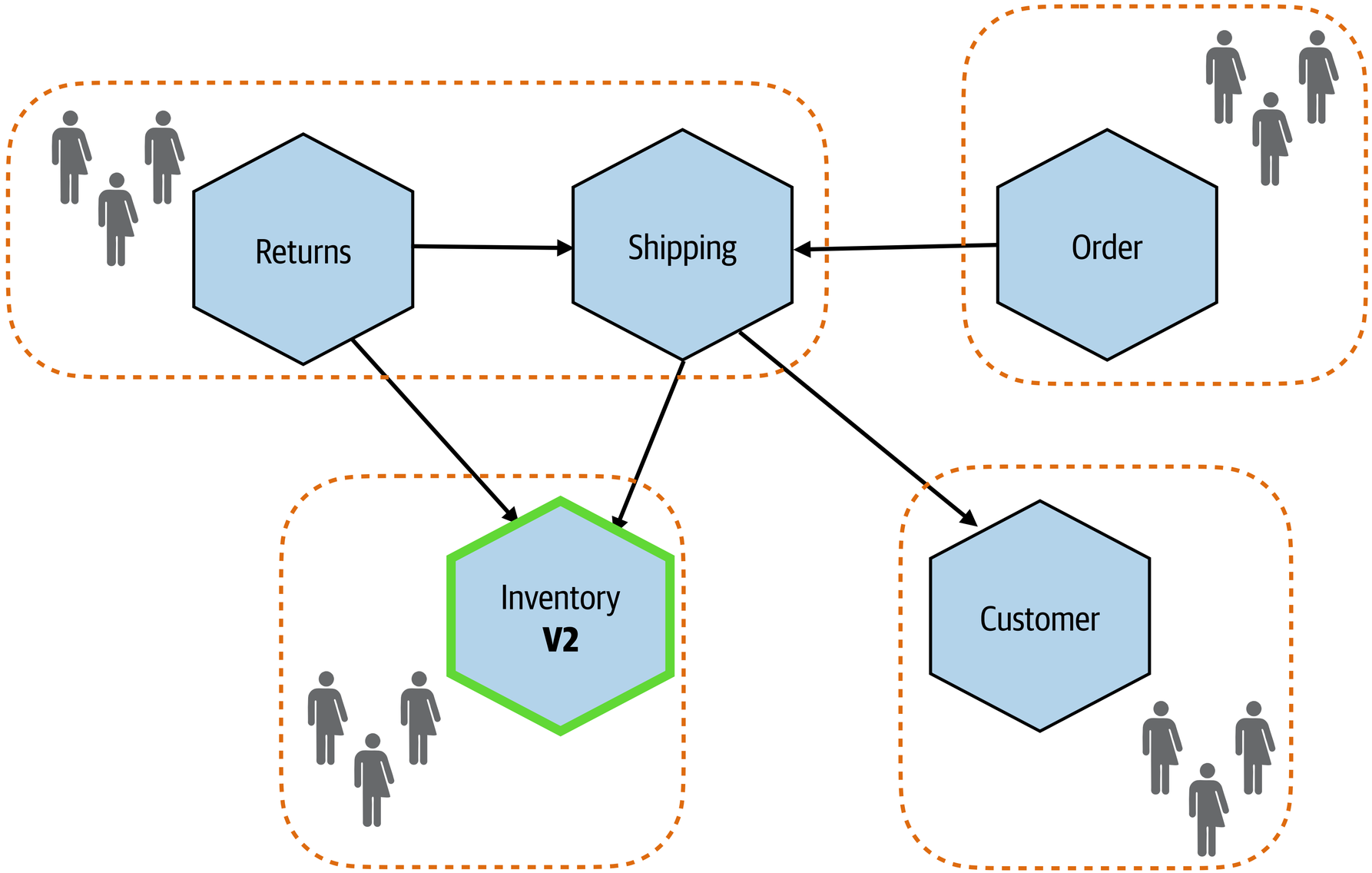
INDEPENDENT DEVELOPMENT & DELIVERY



INDEPENDENT DEVELOPMENT & DELIVERY

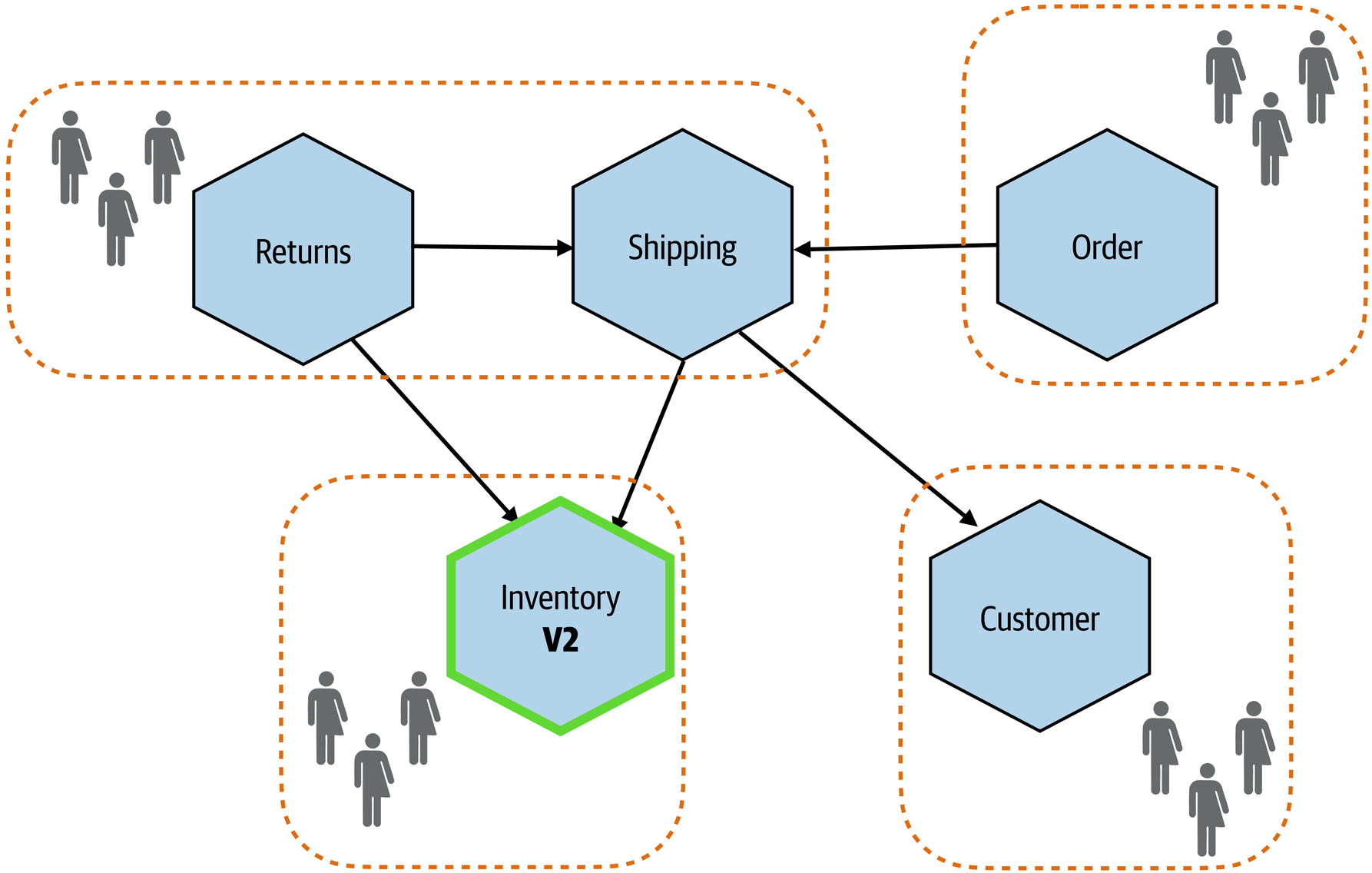


INDEPENDENT DEVELOPMENT & DELIVERY



Reduce release co-ordination

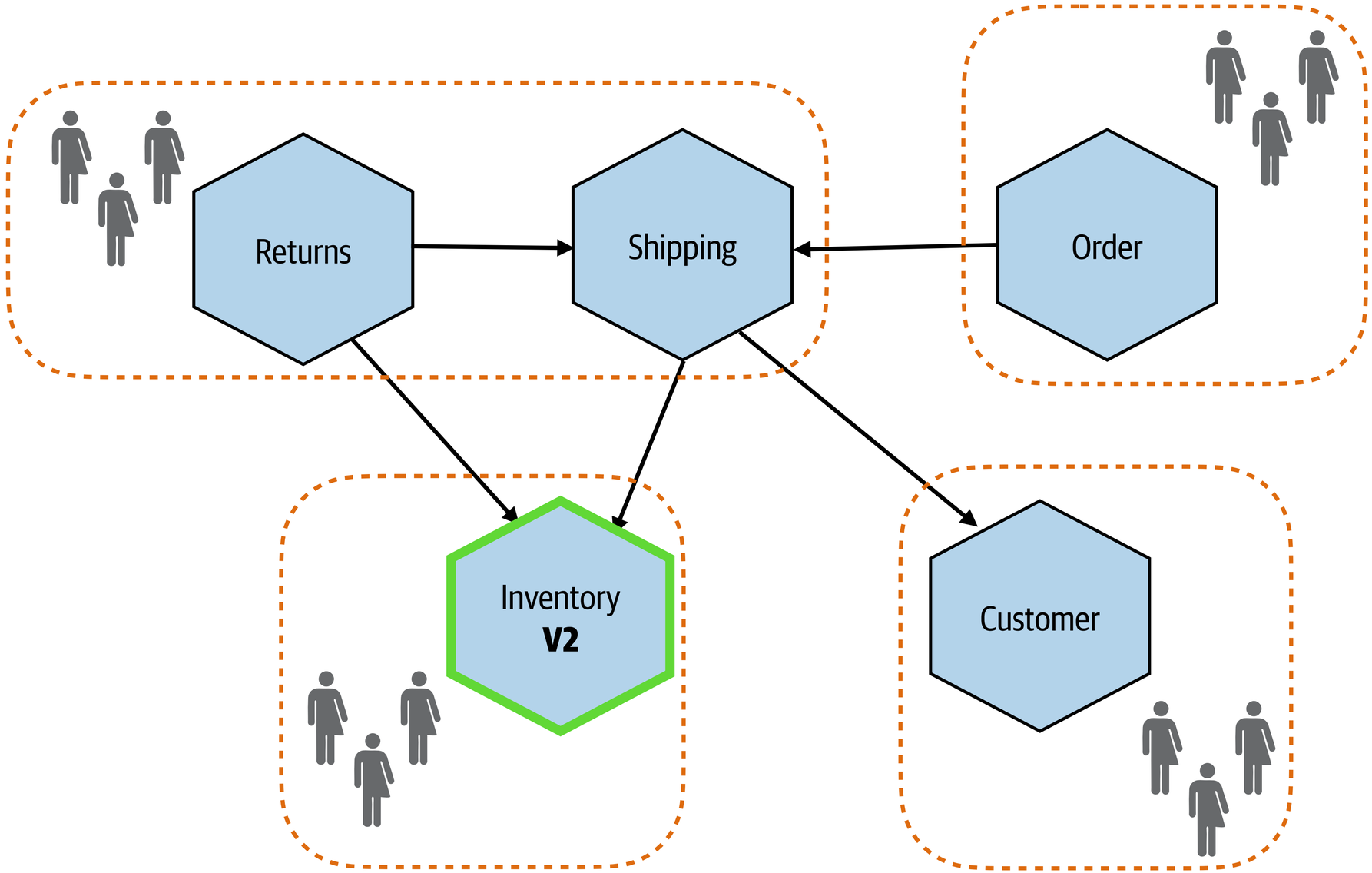
INDEPENDENT DEVELOPMENT & DELIVERY



Reduce release co-ordination

Improve release frequency

INDEPENDENT DEVELOPMENT & DELIVERY

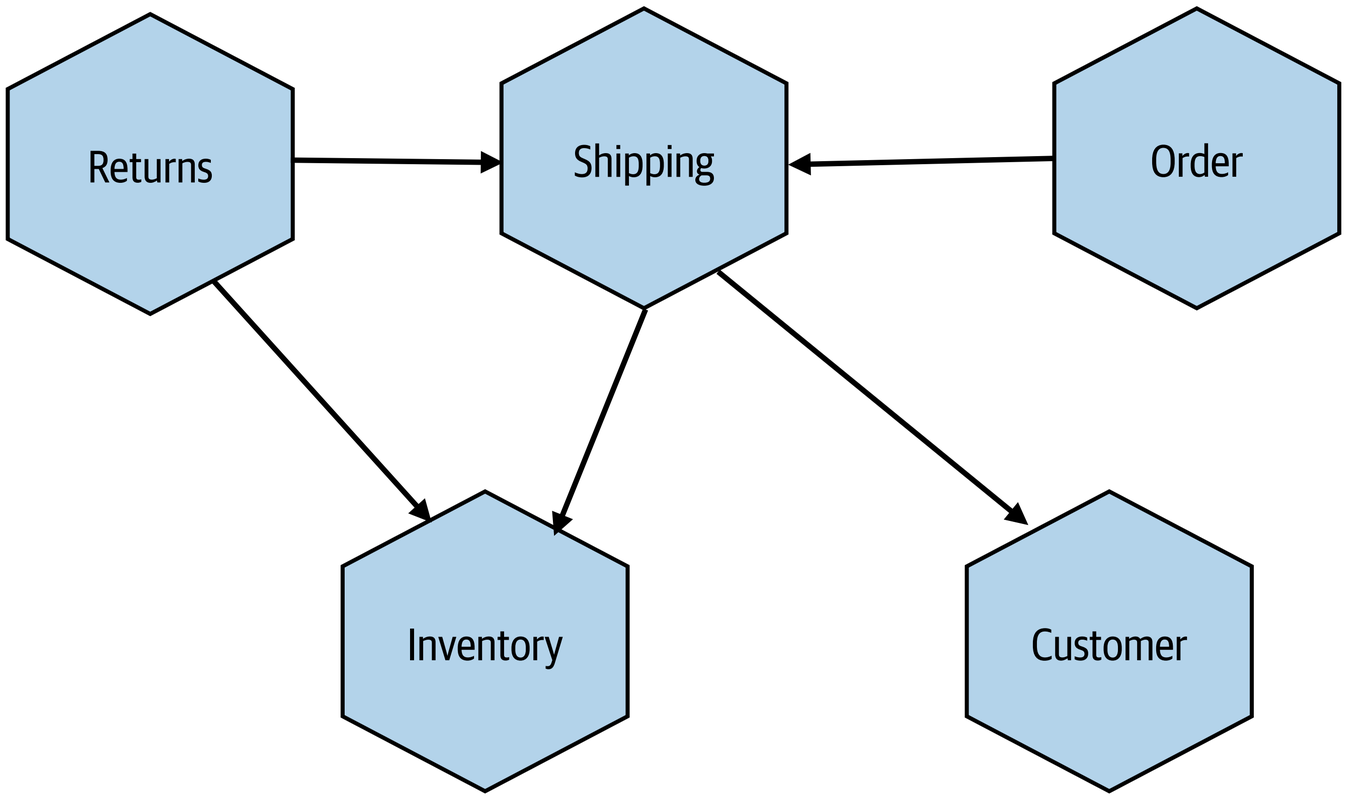


Reduce release co-ordination

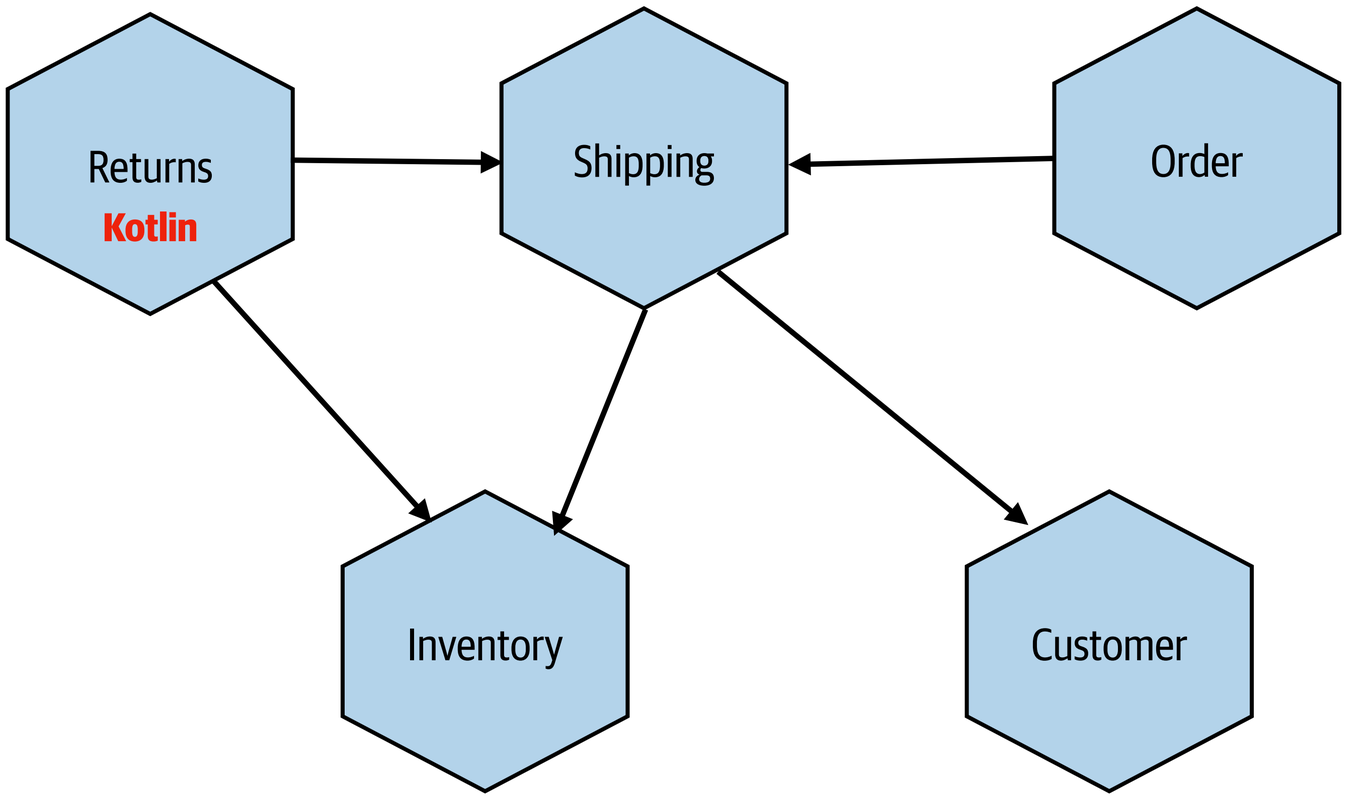
Improve release frequency

Enables full lifecycle ownership

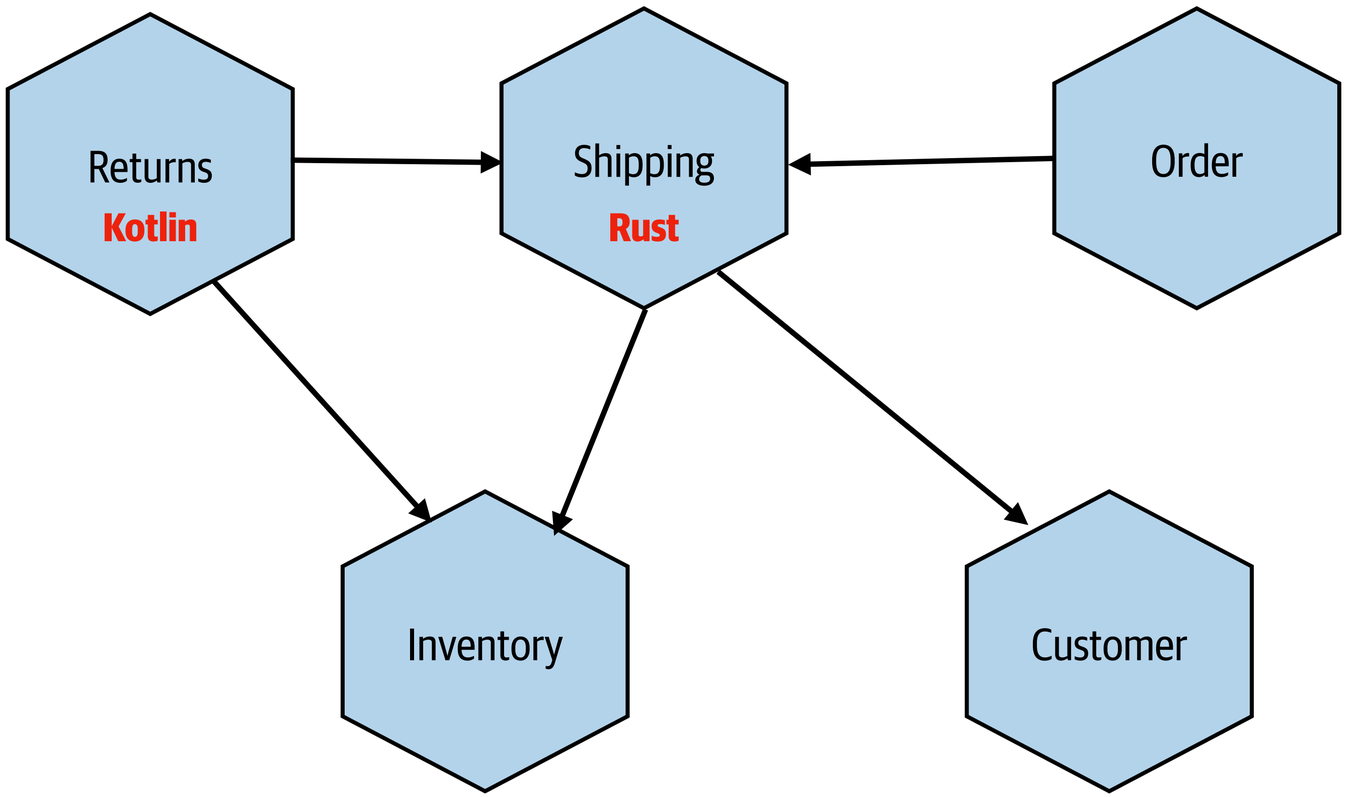
DIFFERENT TECHNOLOGY



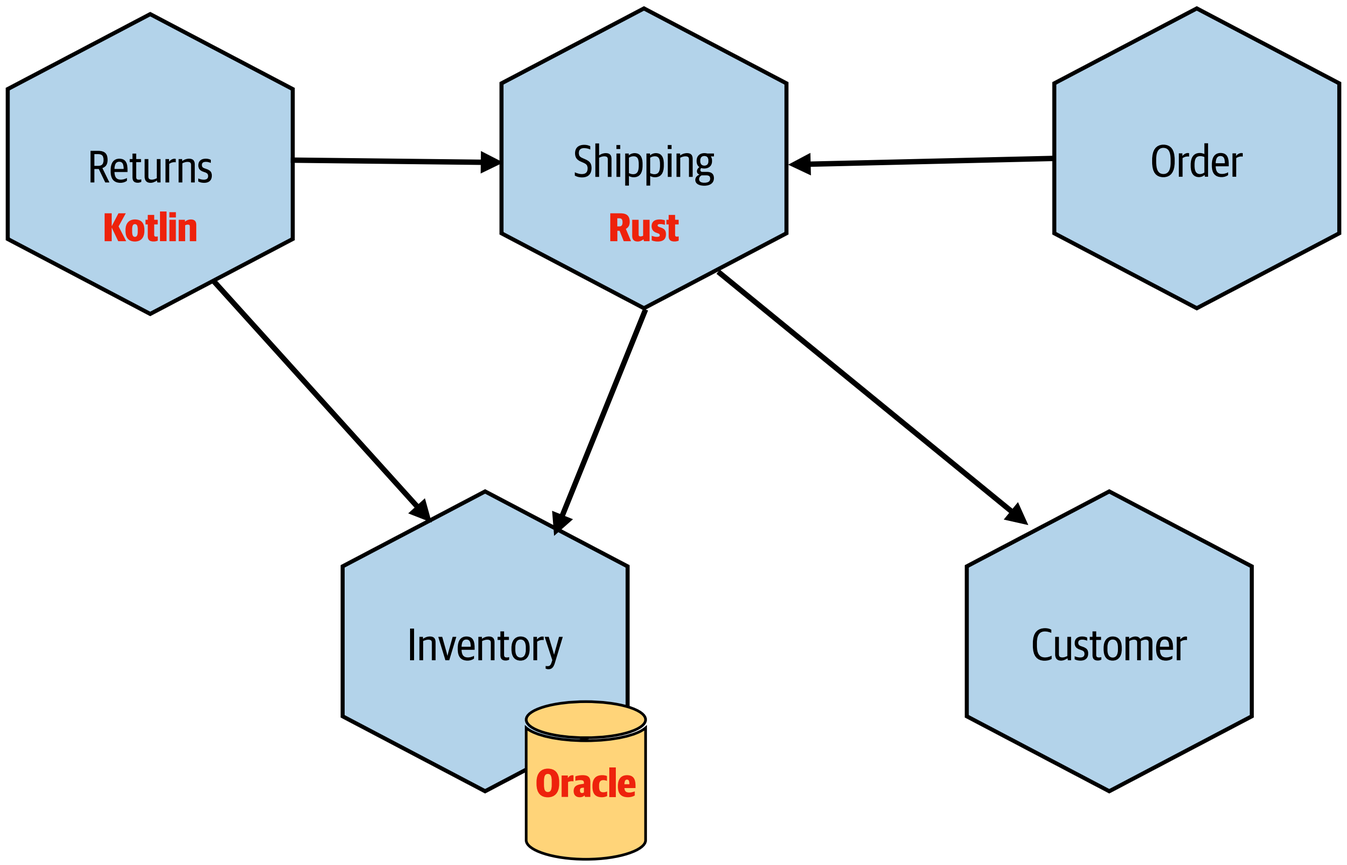
DIFFERENT TECHNOLOGY



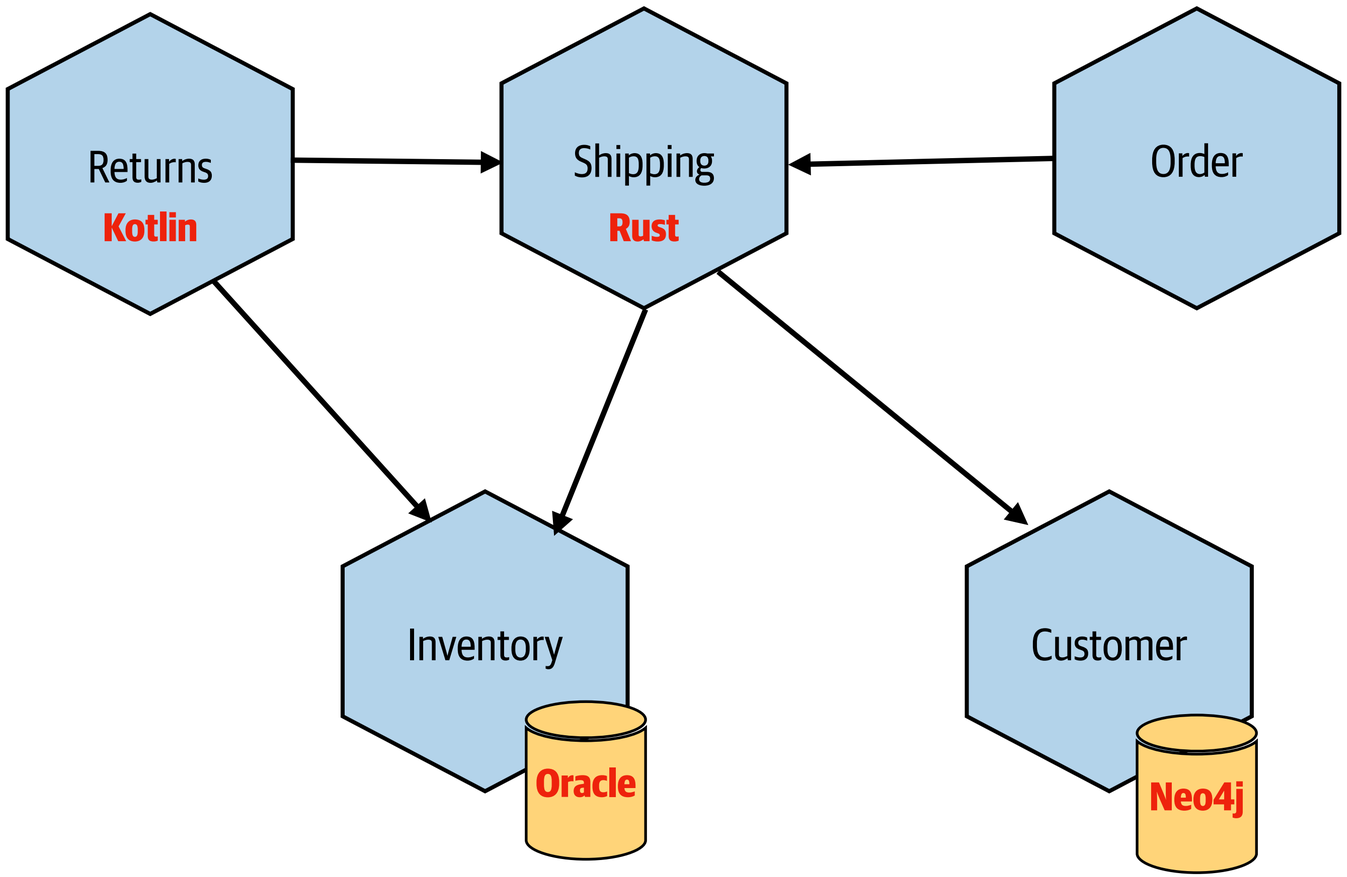
DIFFERENT TECHNOLOGY



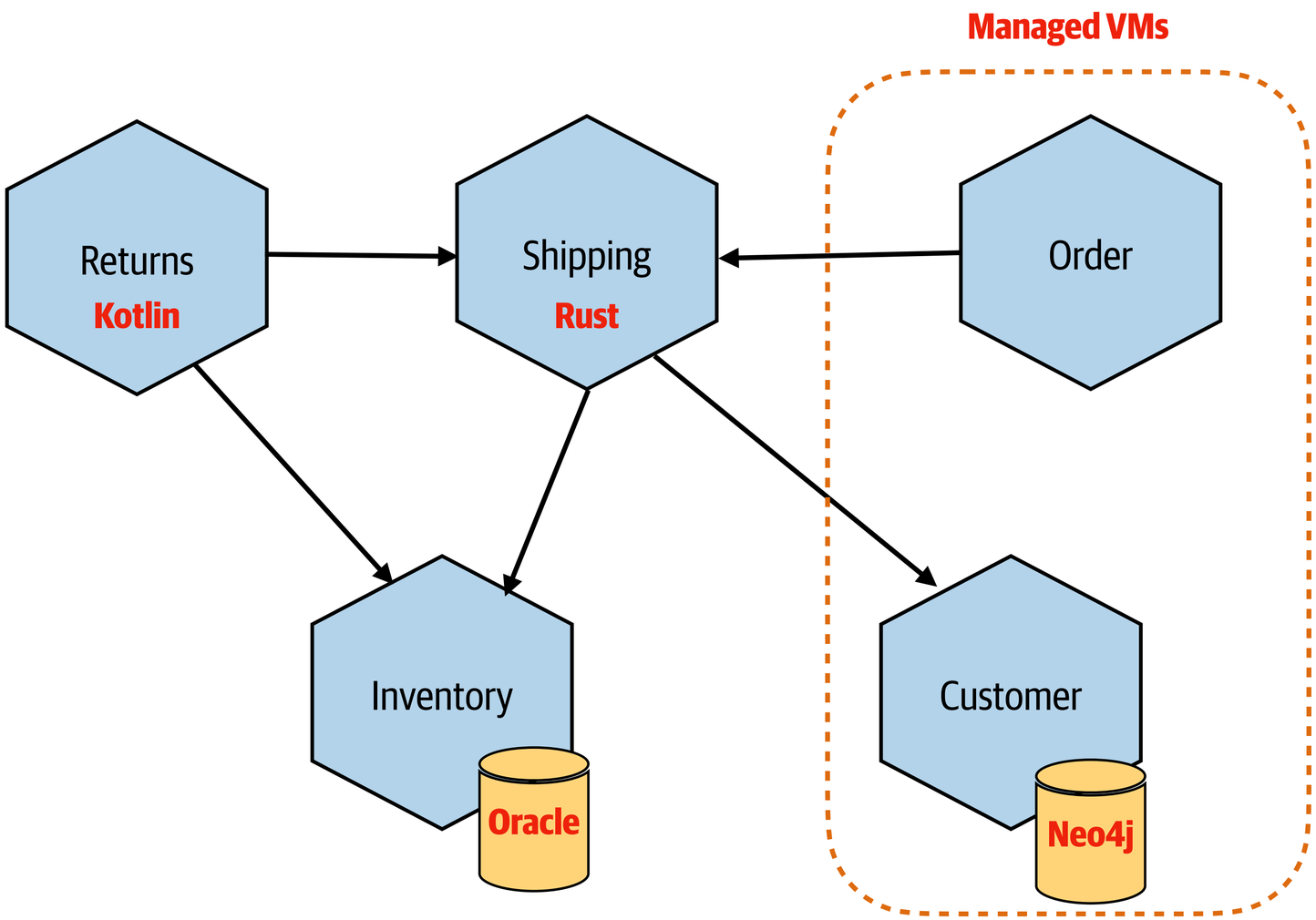
DIFFERENT TECHNOLOGY



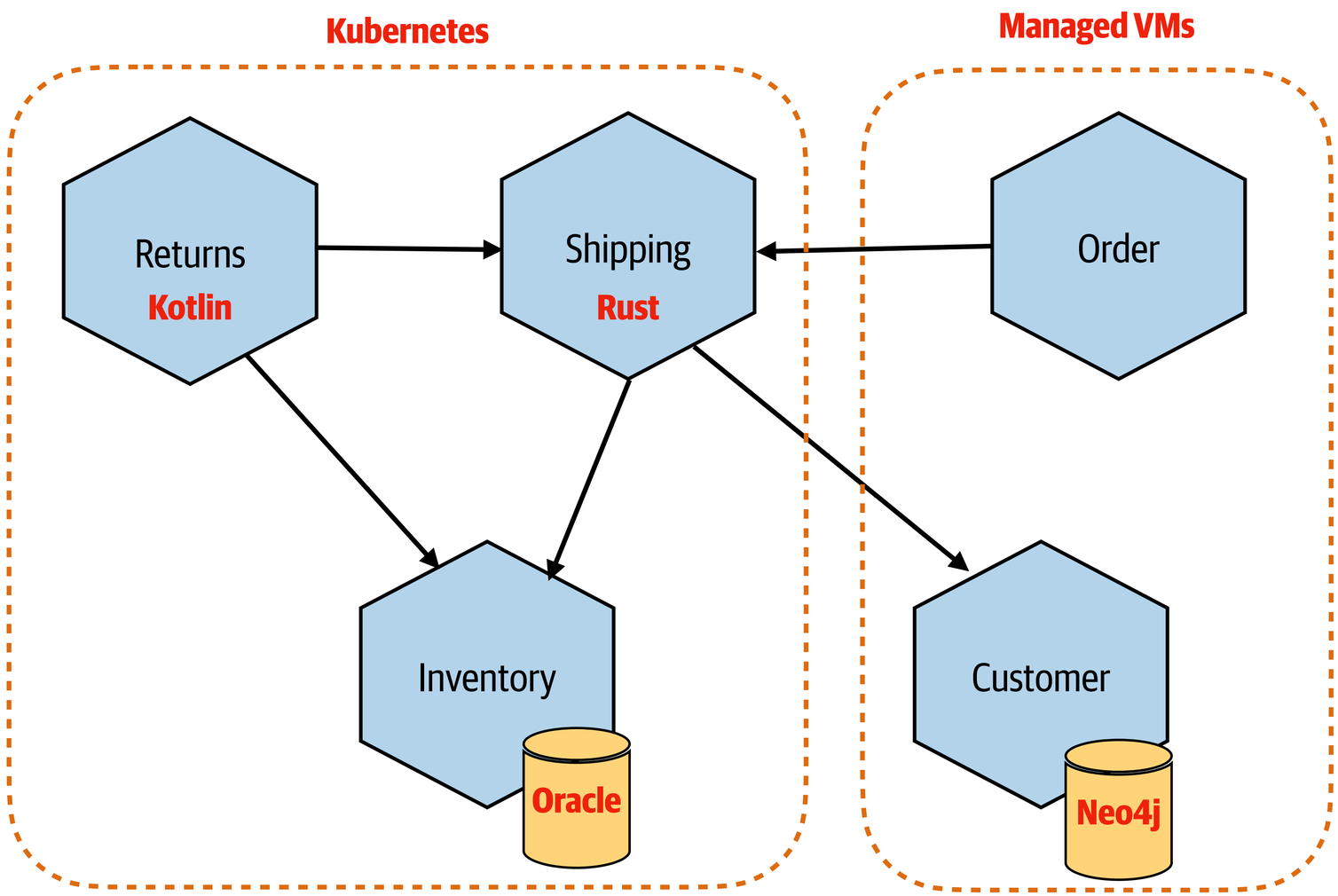
DIFFERENT TECHNOLOGY



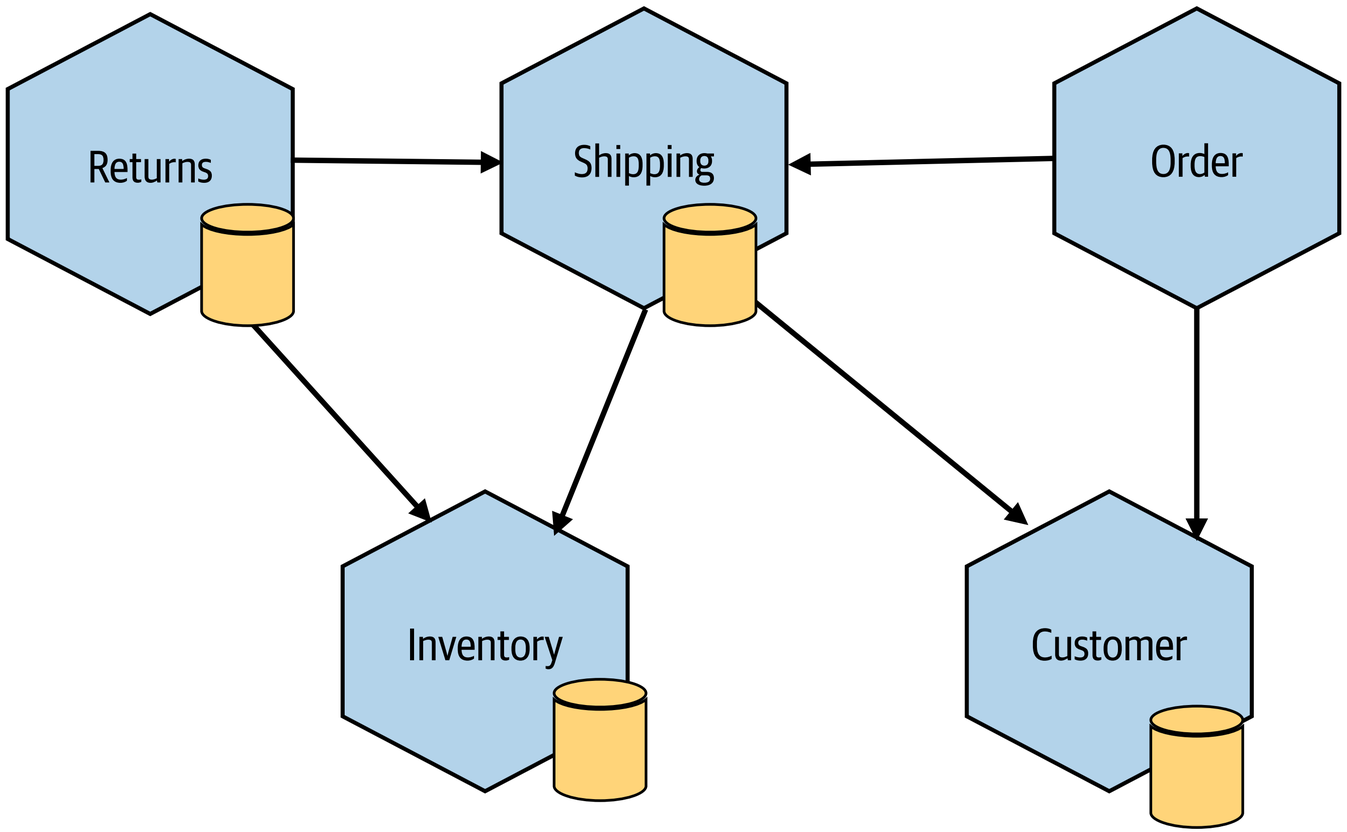
DIFFERENT TECHNOLOGY



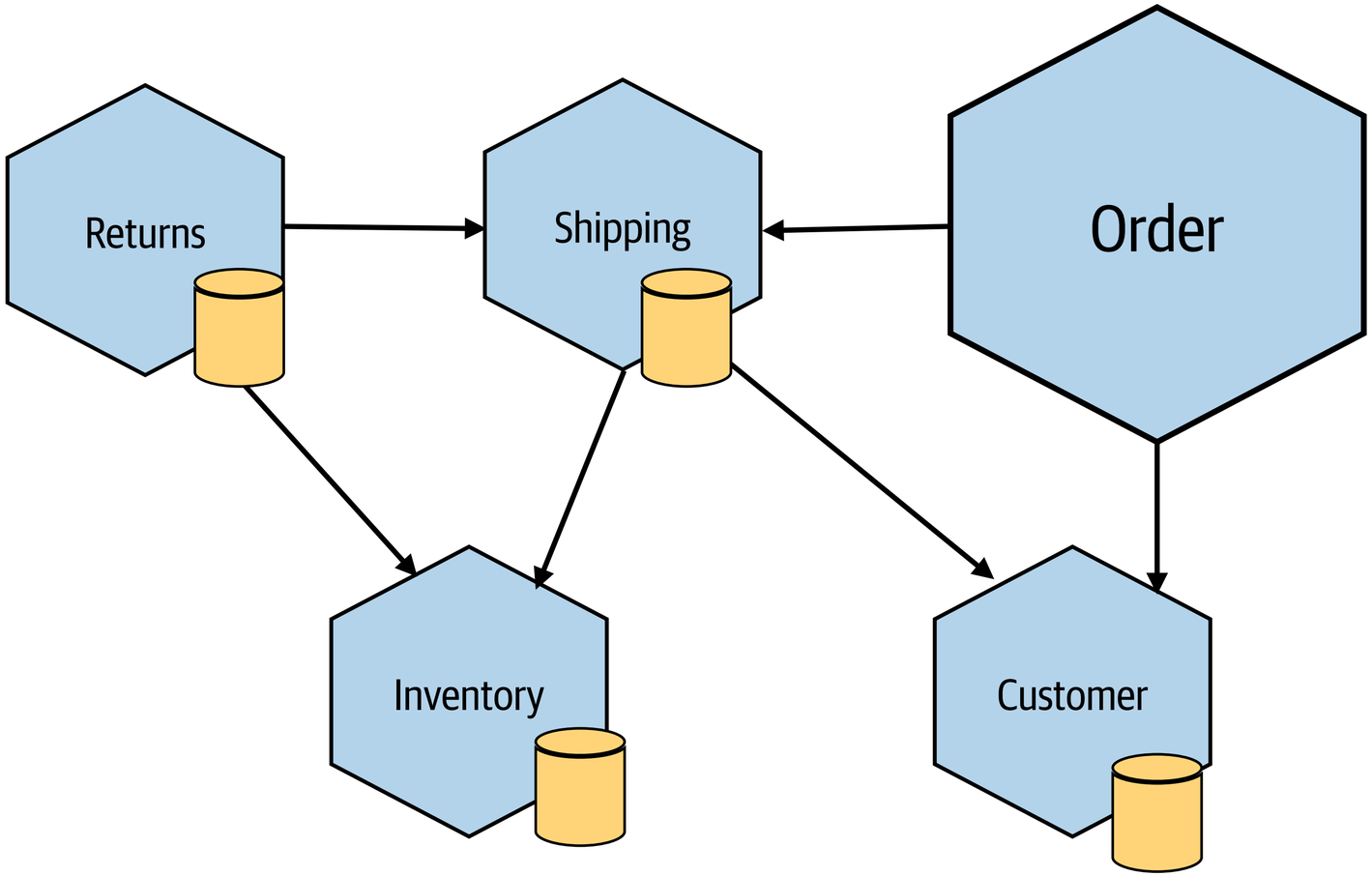
DIFFERENT TECHNOLOGY



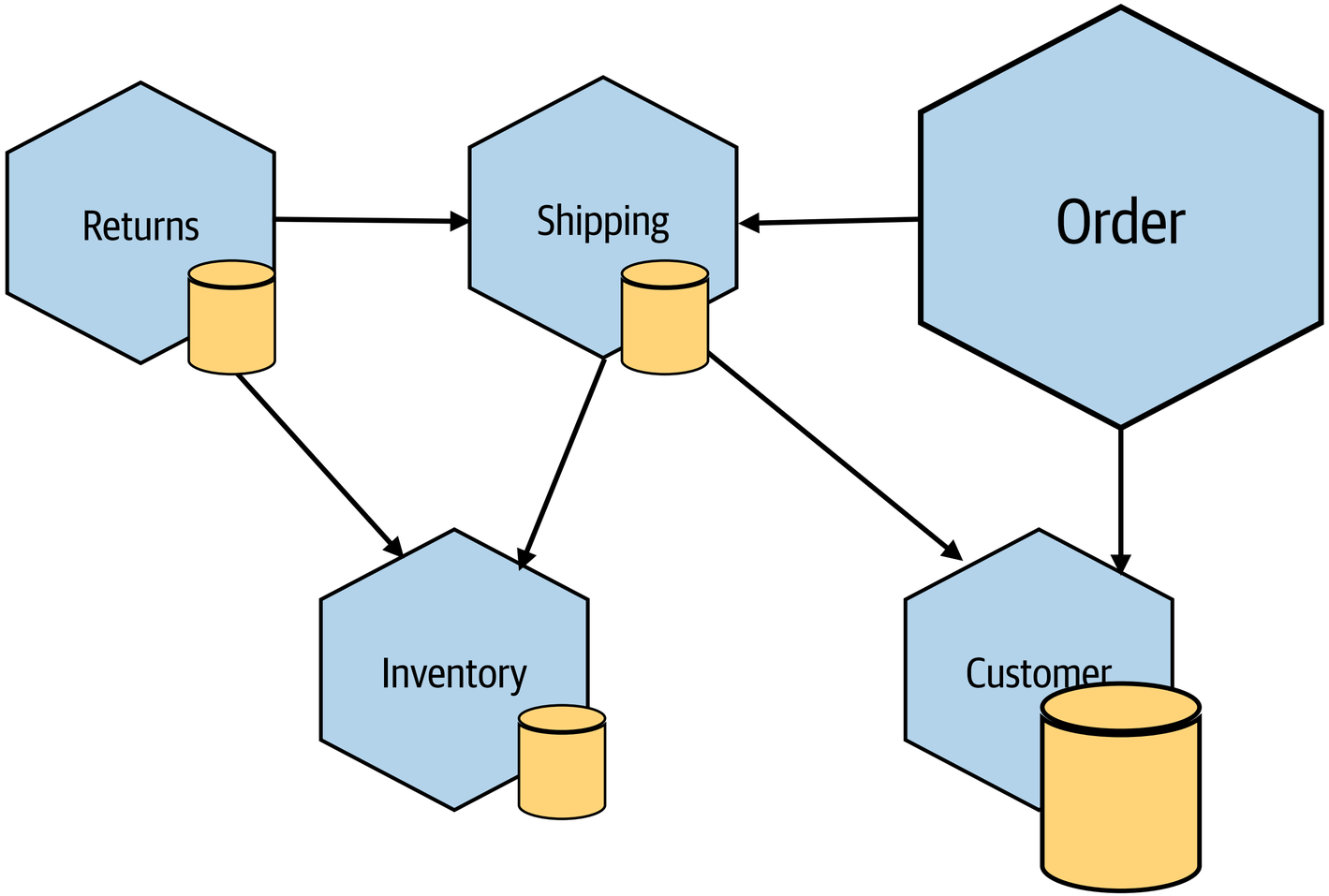
INDEPENDENT SCALING



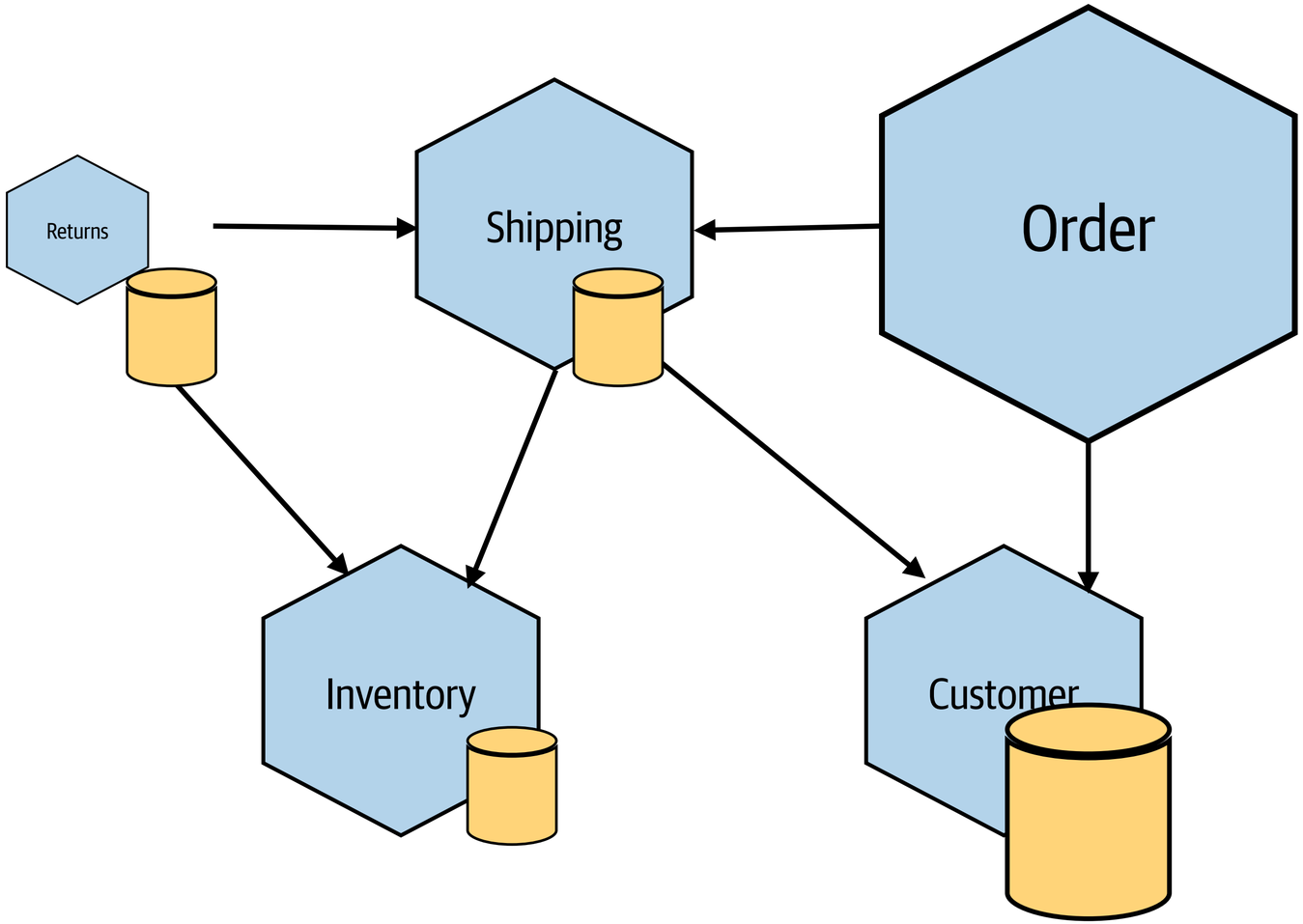
INDEPENDENT SCALING



INDEPENDENT SCALING

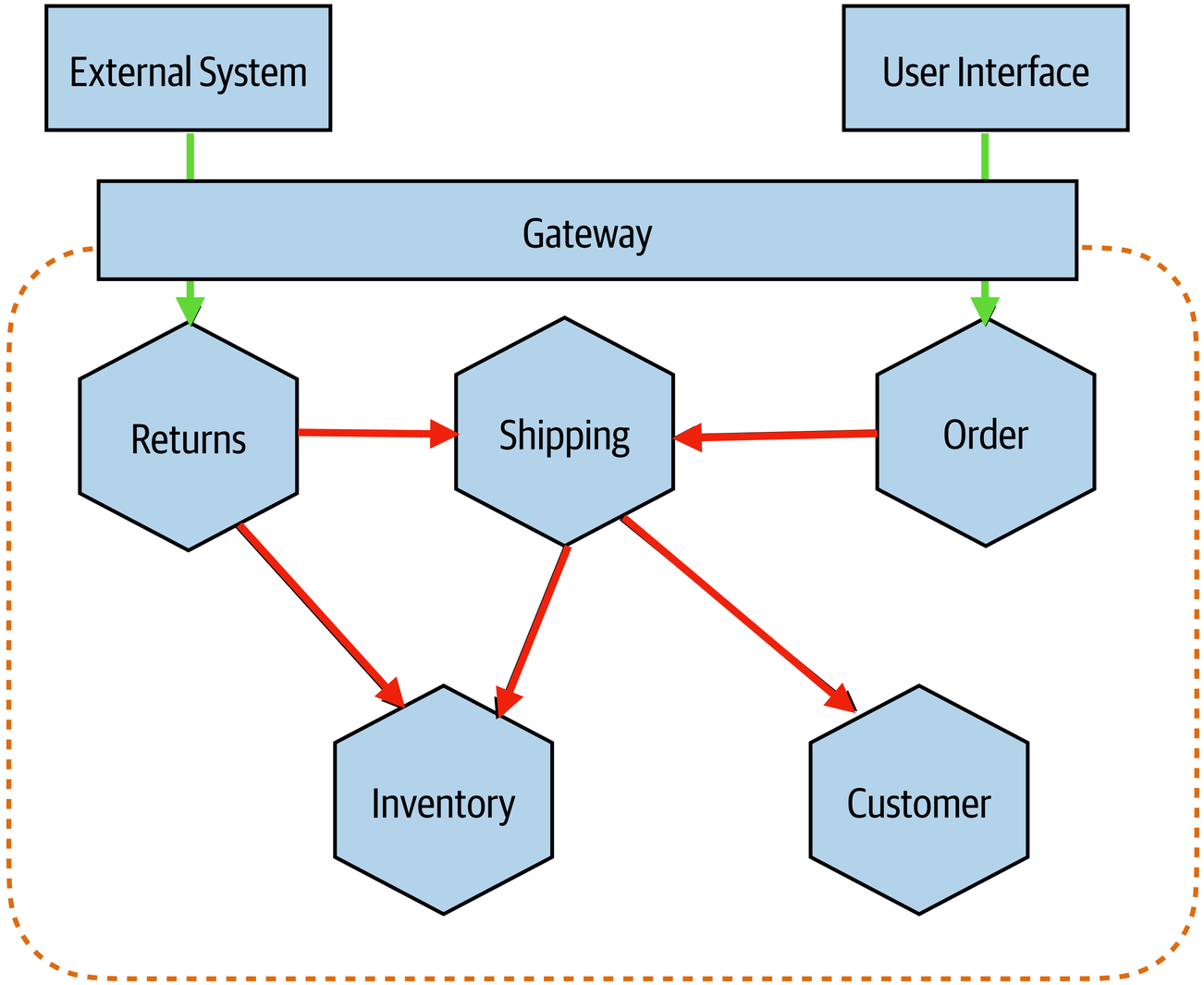


INDEPENDENT SCALING

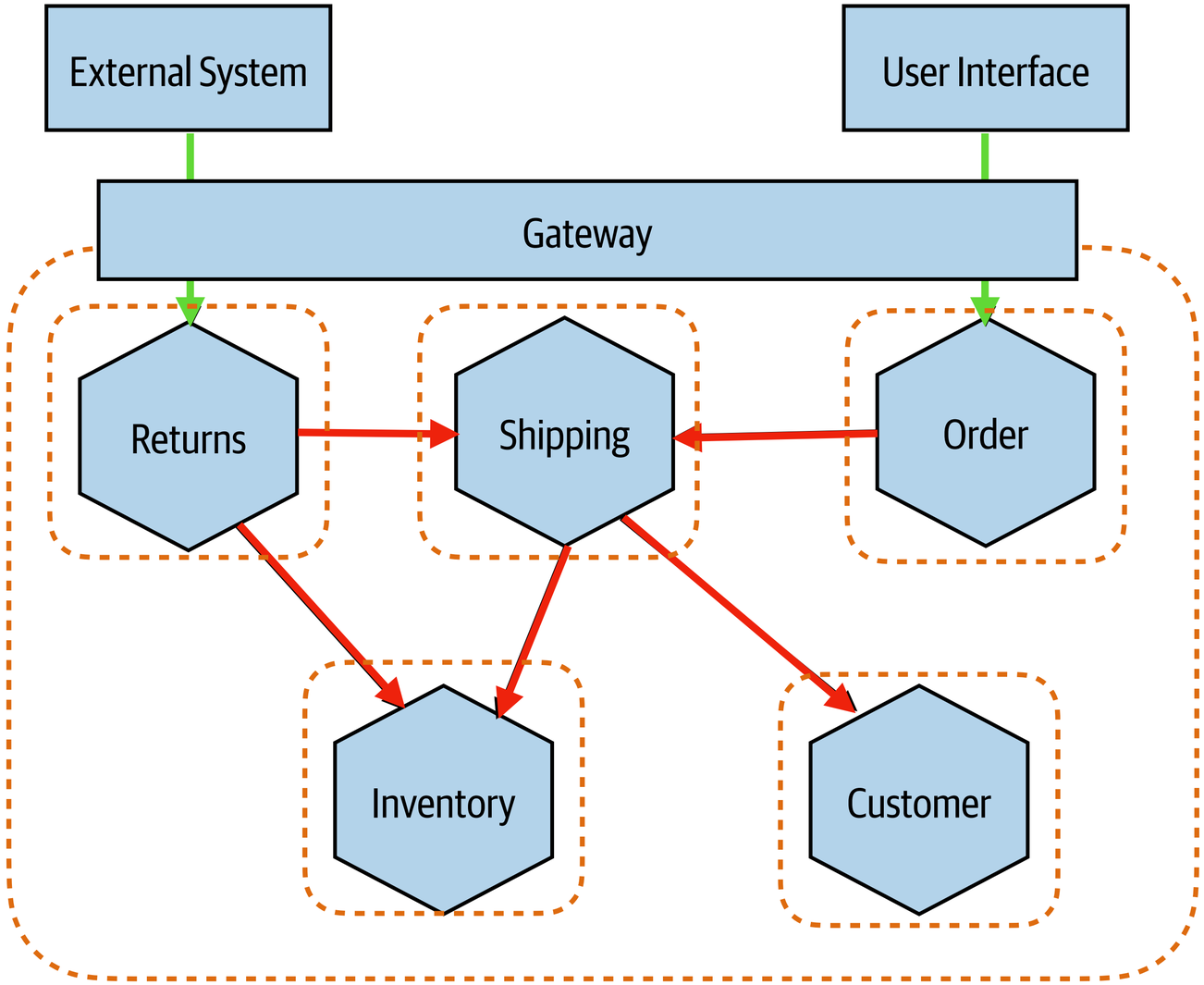


CC Attribution 2.0 Generic https://www.flickr.com/photos/flissphil/52158537/

DEFENCE IN DEPTH



DEFENCE IN DEPTH



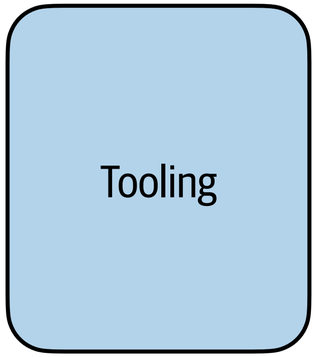
4/4 Reasons not to use microservices!

COGNITIVE OVERLOAD

Based on a model from Adrian Cockcroft:
http://www.slideshare.net/adriancockcroft/evolution-of-microservices-craft-conference

Copyright 2019 Sam Newman & Associates Ltd

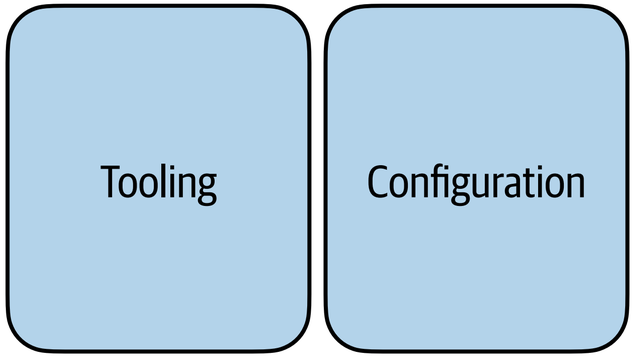
COGNITIVE OVERLOAD



Based on a model from Adrian Cockcroft:
http://www.slideshare.net/adriancockcroft/evolution-of-microservices-craft-conference

Copyright 2019 Sam Newman & Associates Ltd

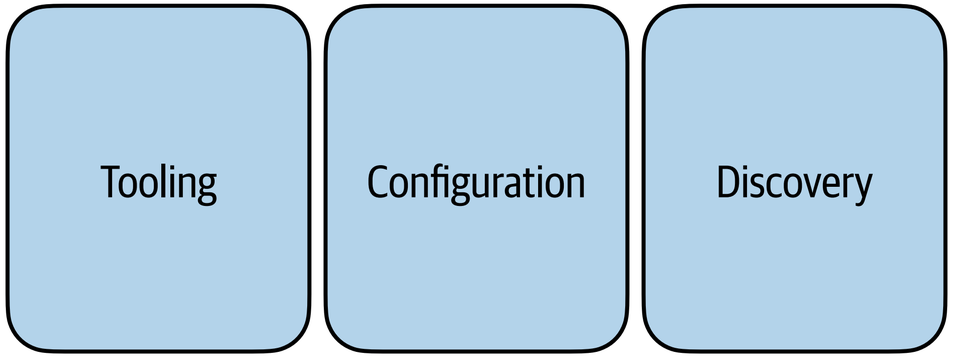
COGNITIVE OVERLOAD



Based on a model from Adrian Cockcroft:
http://www.slideshare.net/adriancockcroft/evolution-of-microservices-craft-conference

Copyright 2019 Sam Newman & Associates Ltd

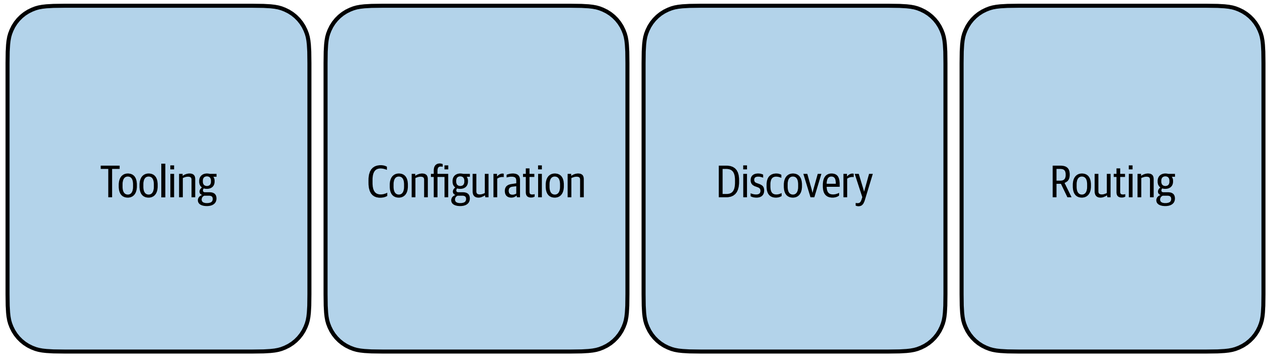
COGNITIVE OVERLOAD



Based on a model from Adrian Cockcroft:
http://www.slideshare.net/adriancockcroft/evolution-of-microservices-craft-conference

Copyright 2019 Sam Newman & Associates Ltd

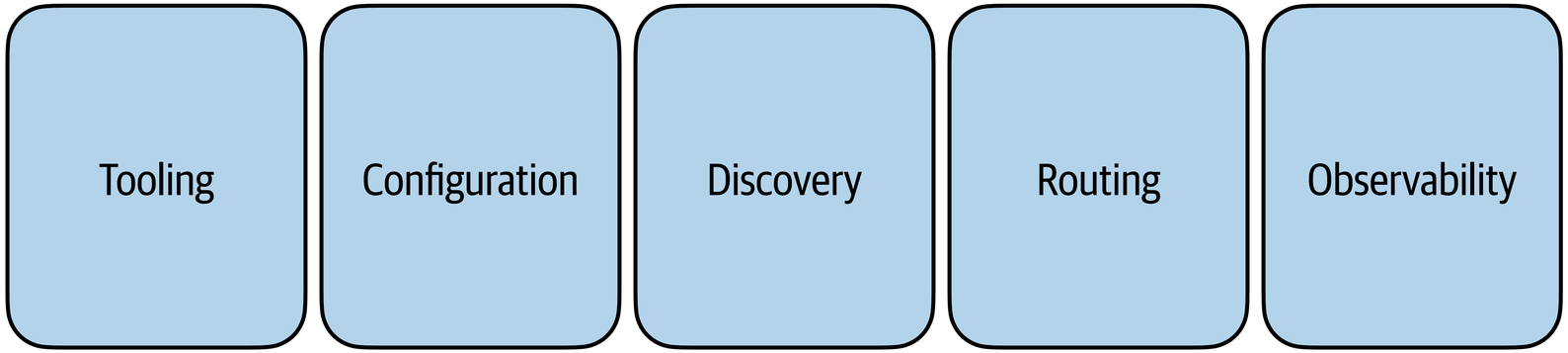
COGNITIVE OVERLOAD



Based on a model from Adrian Cockcroft:
http://www.slideshare.net/adriancockcroft/evolution-of-microservices-craft-conference

Copyright 2019 Sam Newman & Associates Ltd

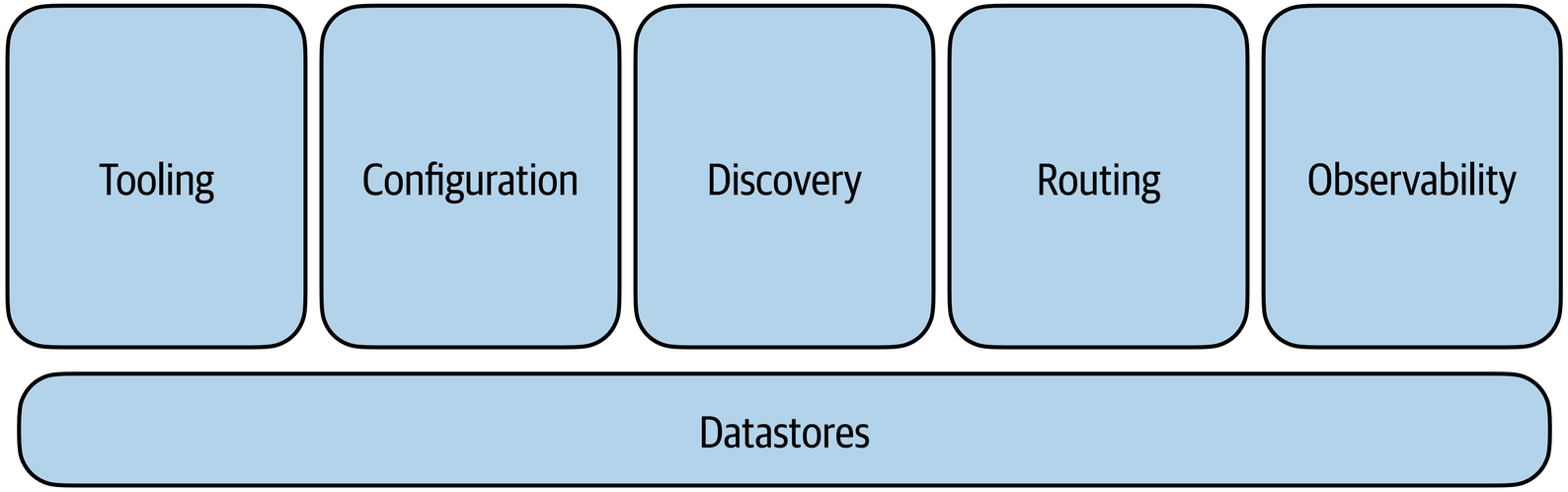
COGNITIVE OVERLOAD



Based on a model from Adrian Cockcroft:
http://www.slideshare.net/adriancockcroft/evolution-of-microservices-craft-conference

Copyright 2019 Sam Newman & Associates Ltd

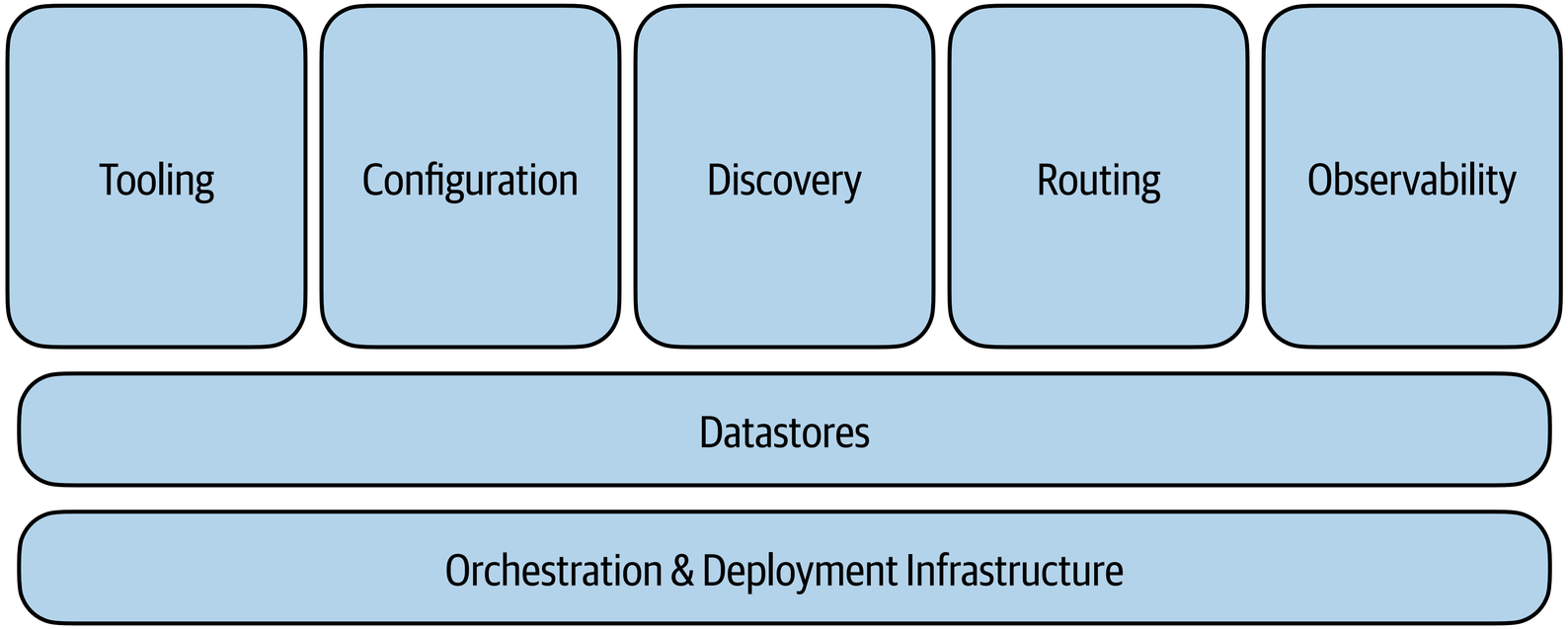
COGNITIVE OVERLOAD



Based on a model from Adrian Cockcroft:
http://www.slideshare.net/adriancockcroft/evolution-of-microservices-craft-conference

Copyright 2019 Sam Newman & Associates Ltd

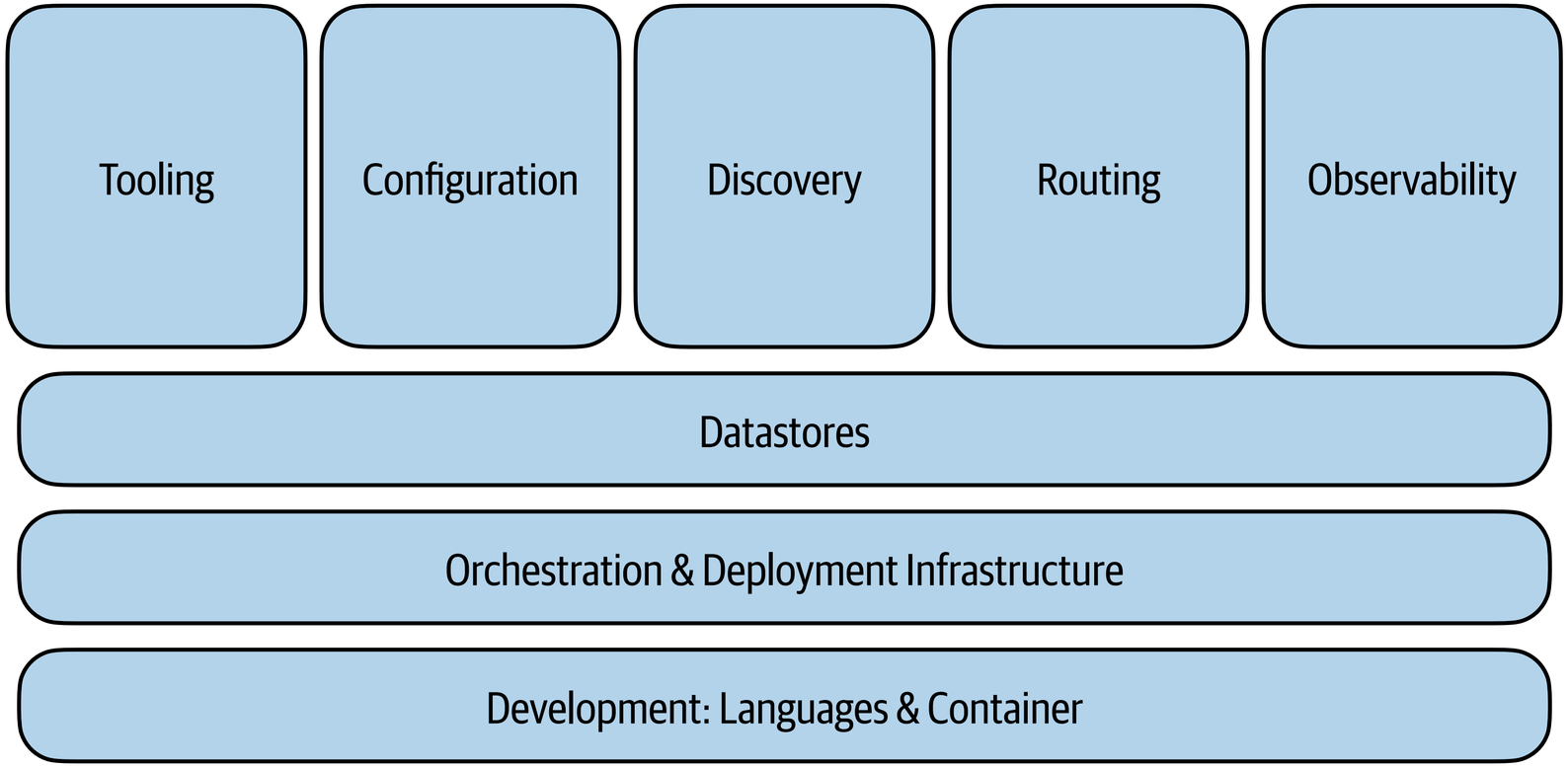
COGNITIVE OVERLOAD



Based on a model from Adrian Cockcroft:
http://www.slideshare.net/adriancockcroft/evolution-of-microservices-craft-conference

Copyright 2019 Sam Newman & Associates Ltd

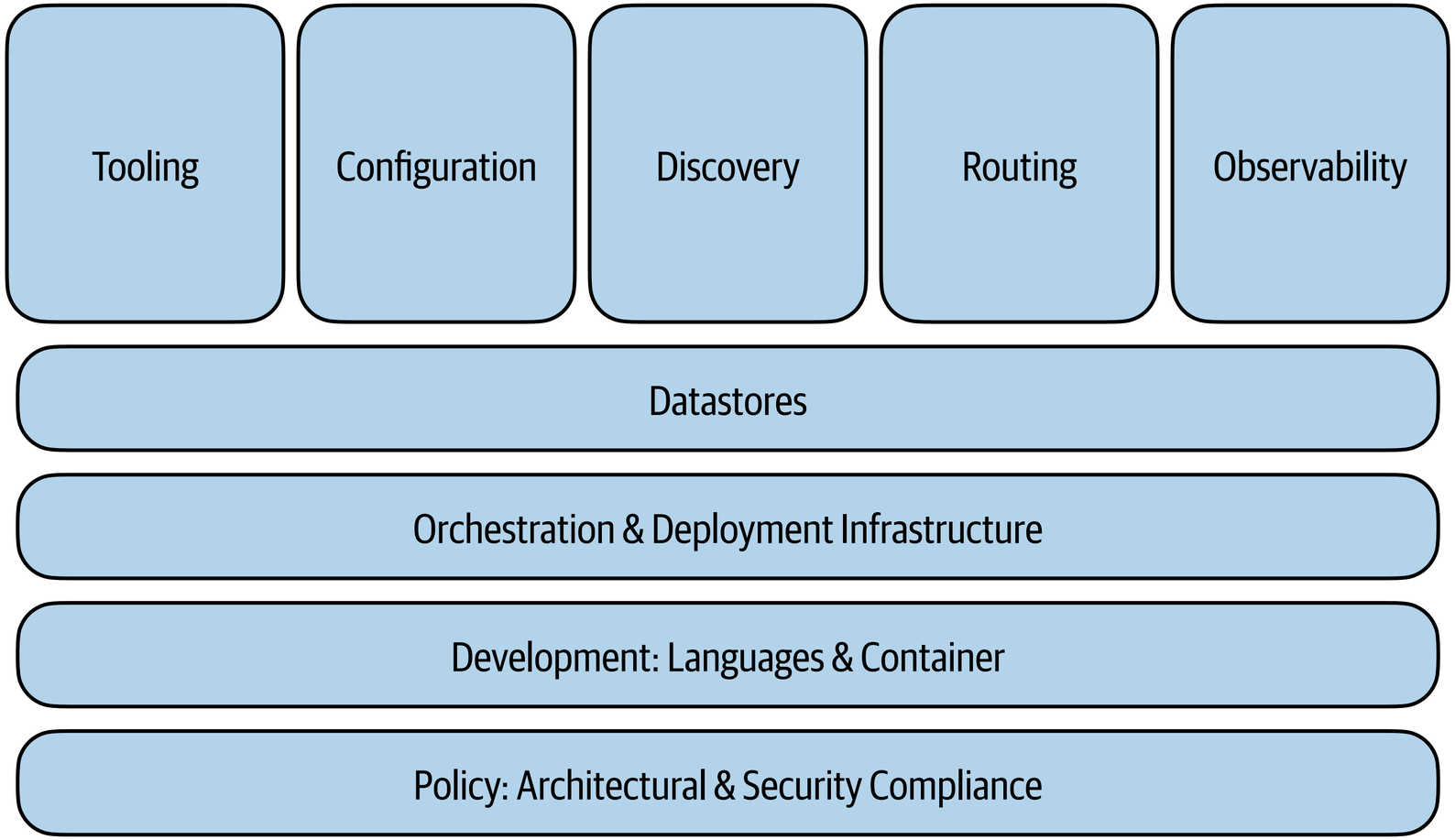
COGNITIVE OVERLOAD



Based on a model from Adrian Cockcroft:
http://www.slideshare.net/adriancockcroft/evolution-of-microservices-craft-conference

Copyright 2019 Sam Newman & Associates Ltd

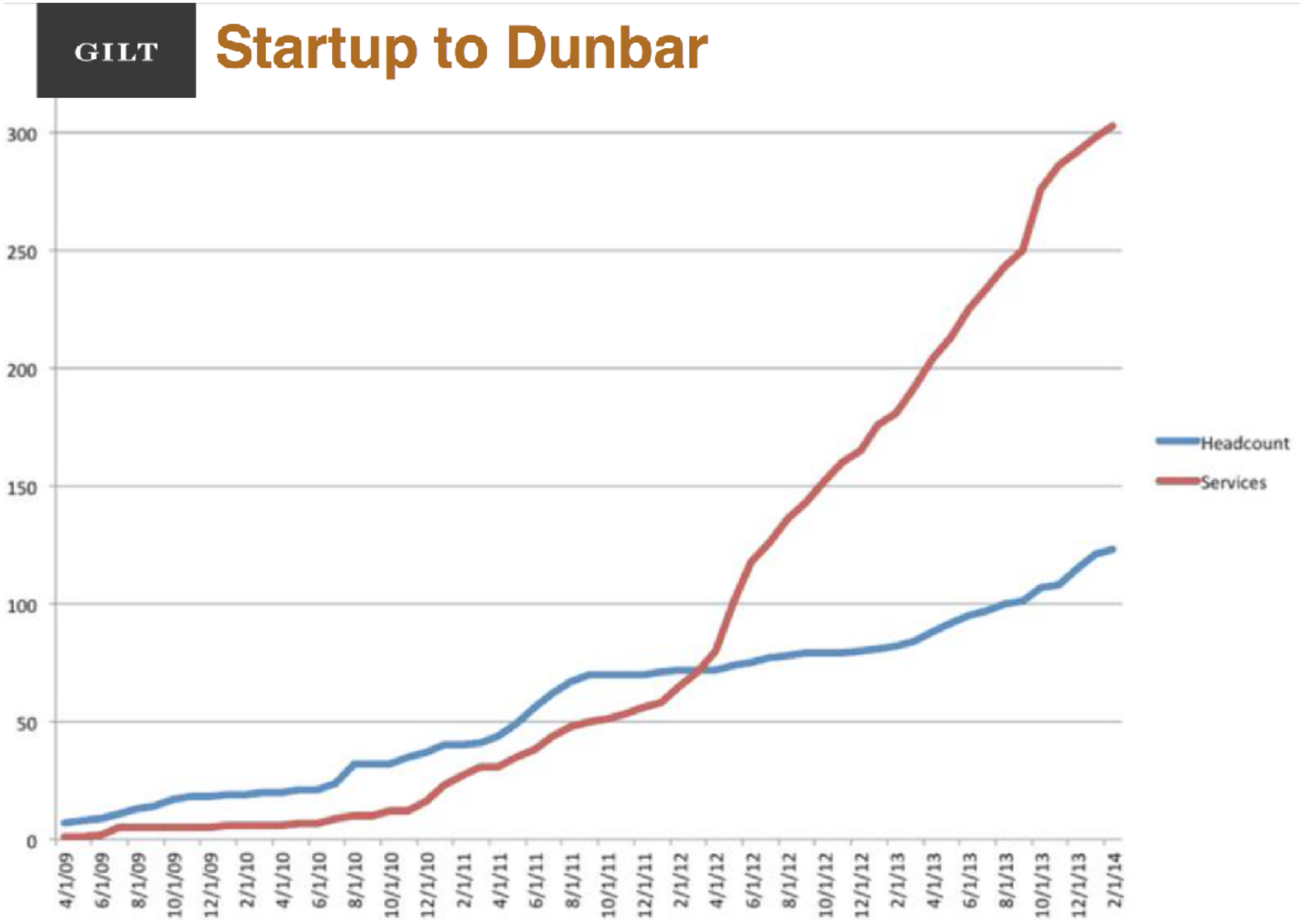
COGNITIVE OVERLOAD



Based on a model from Adrian Cockcroft:
http://www.slideshare.net/adriancockcroft/evolution-of-microservices-craft-conference

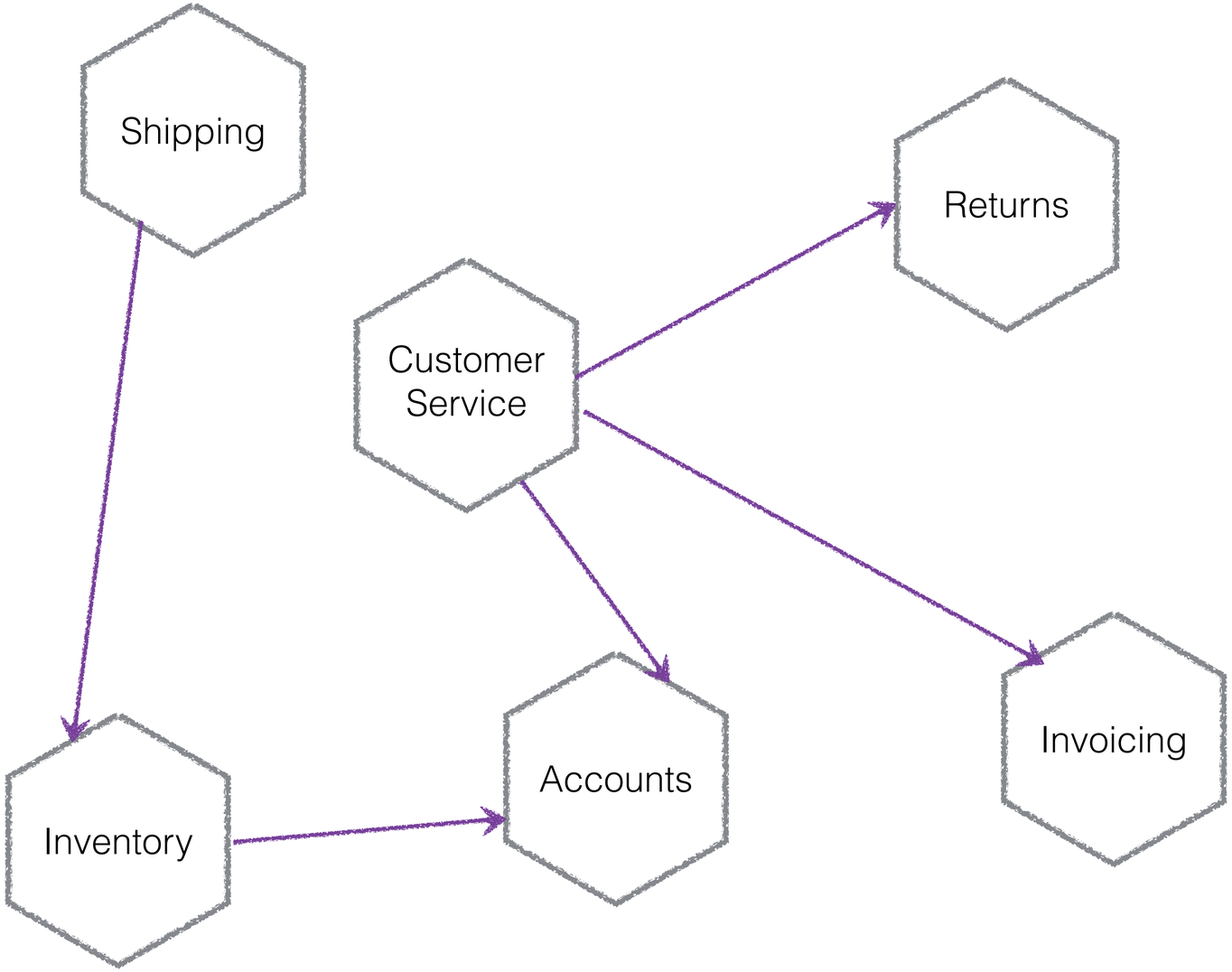
Copyright 2019 Sam Newman & Associates Ltd

IT TAKES TIME…



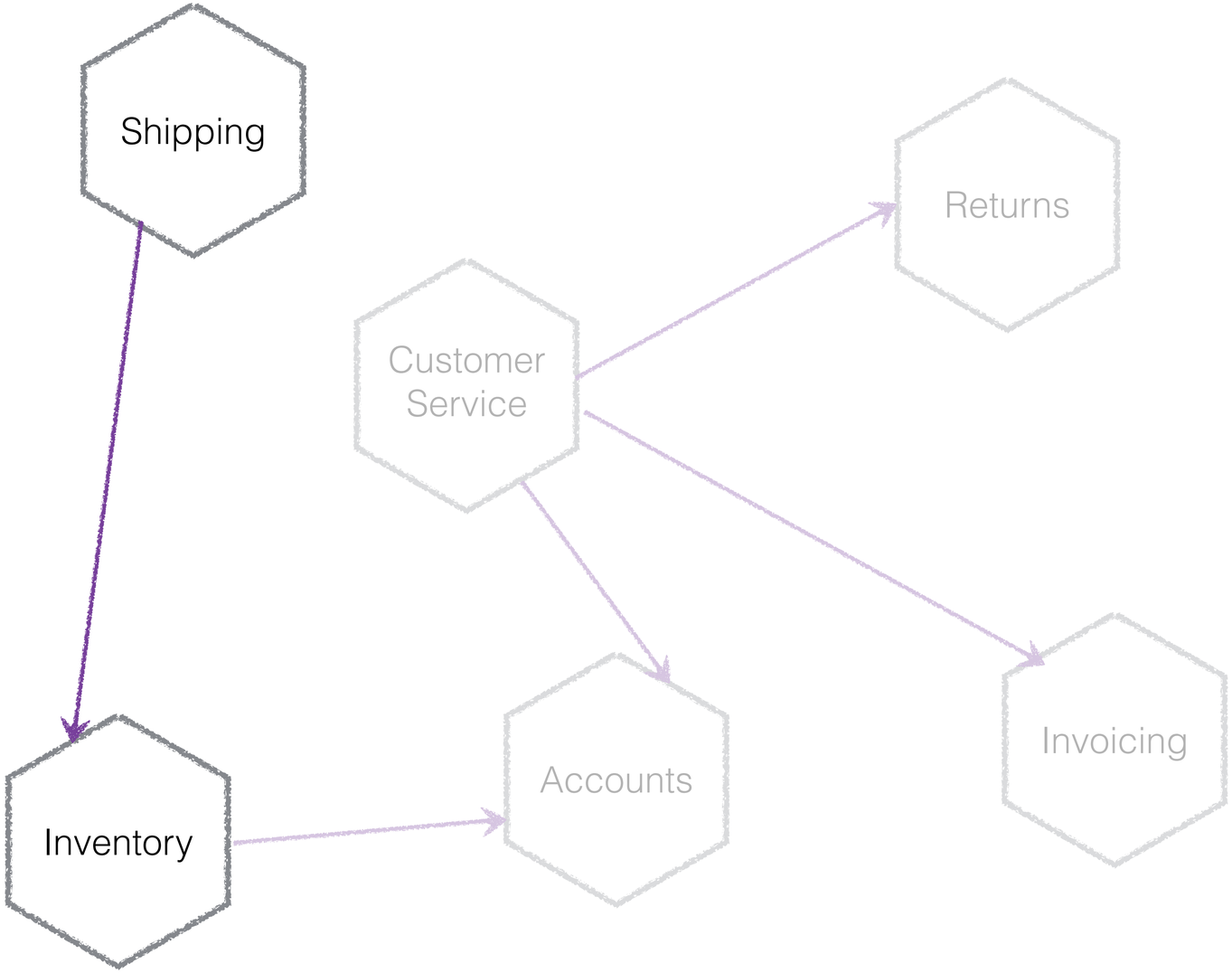
http://www.infoq.com/presentations/scaling-gilt

CROSS-SERVICE TESTING IS MORE INVOLVED



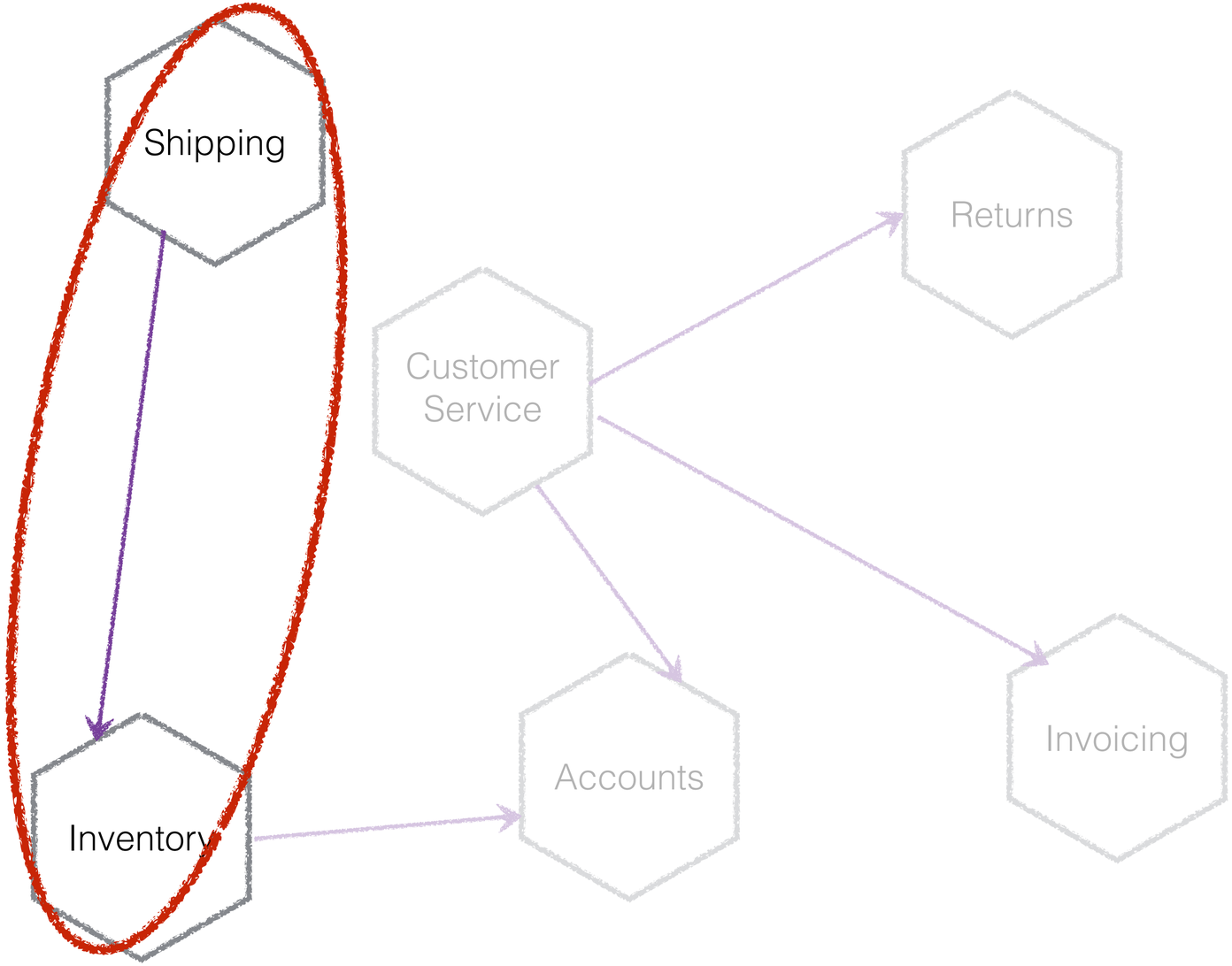
Copyright 2019 Sam Newman & Associates Ltd

CROSS-SERVICE TESTING IS MORE INVOLVED



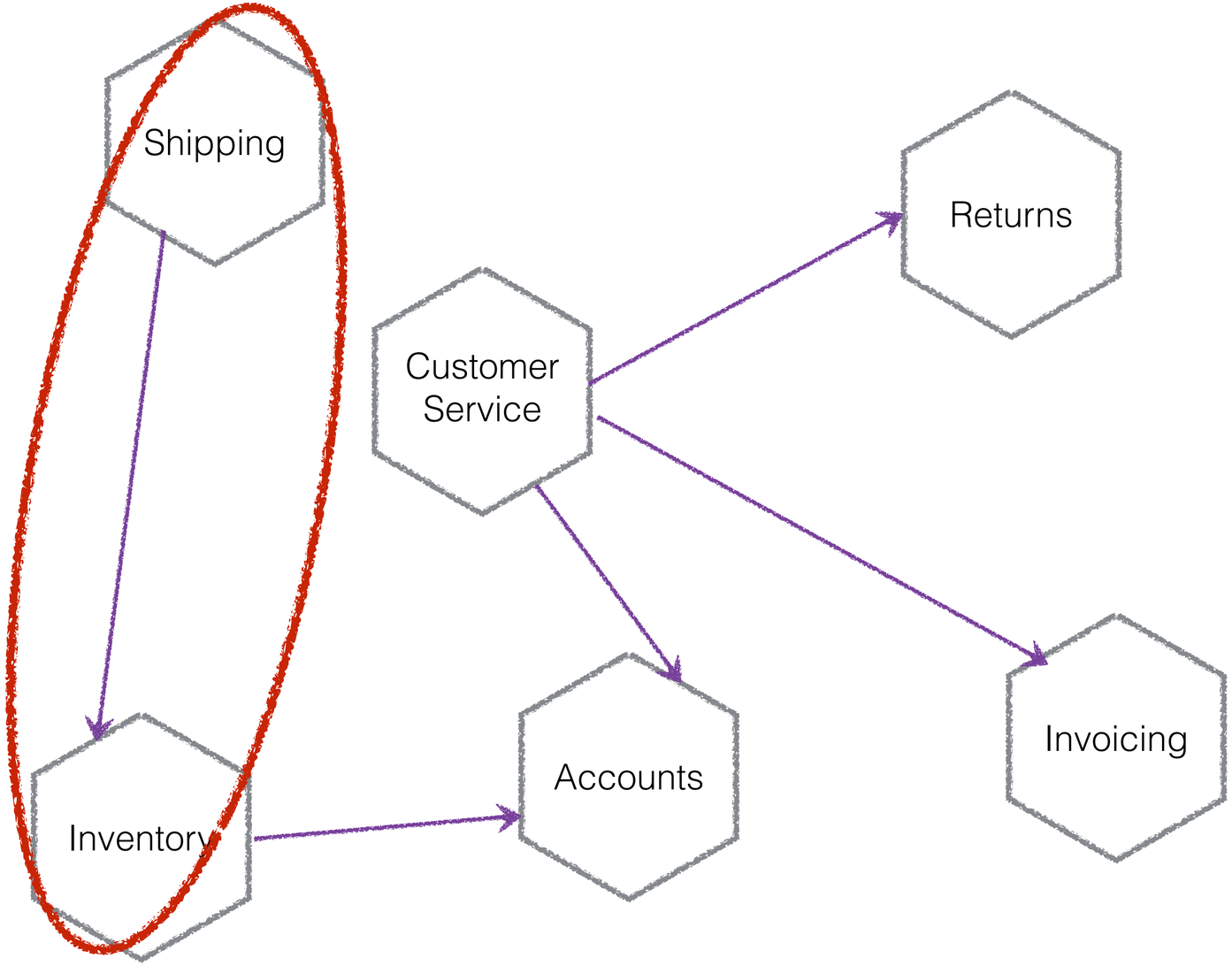
Copyright 2019 Sam Newman & Associates Ltd

CROSS-SERVICE TESTING IS MORE INVOLVED



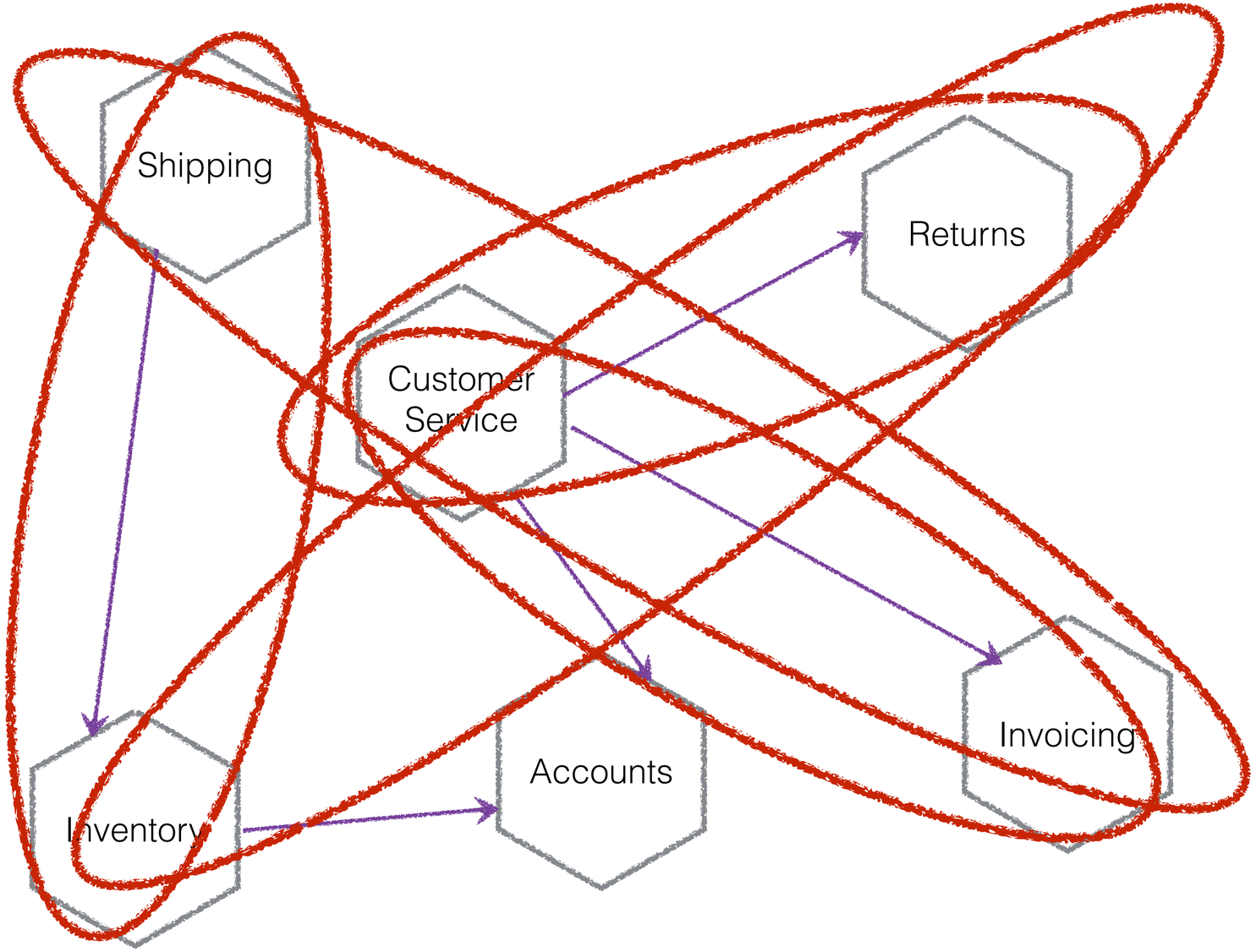
Copyright 2019 Sam Newman & Associates Ltd

CROSS-SERVICE TESTING IS MORE INVOLVED



Copyright 2019 Sam Newman & Associates Ltd

CROSS-SERVICE TESTING IS MORE INVOLVED



Copyright 2019 Sam Newman & Associates Ltd

WHAT’S WRONG?

WHAT’S WRONG?

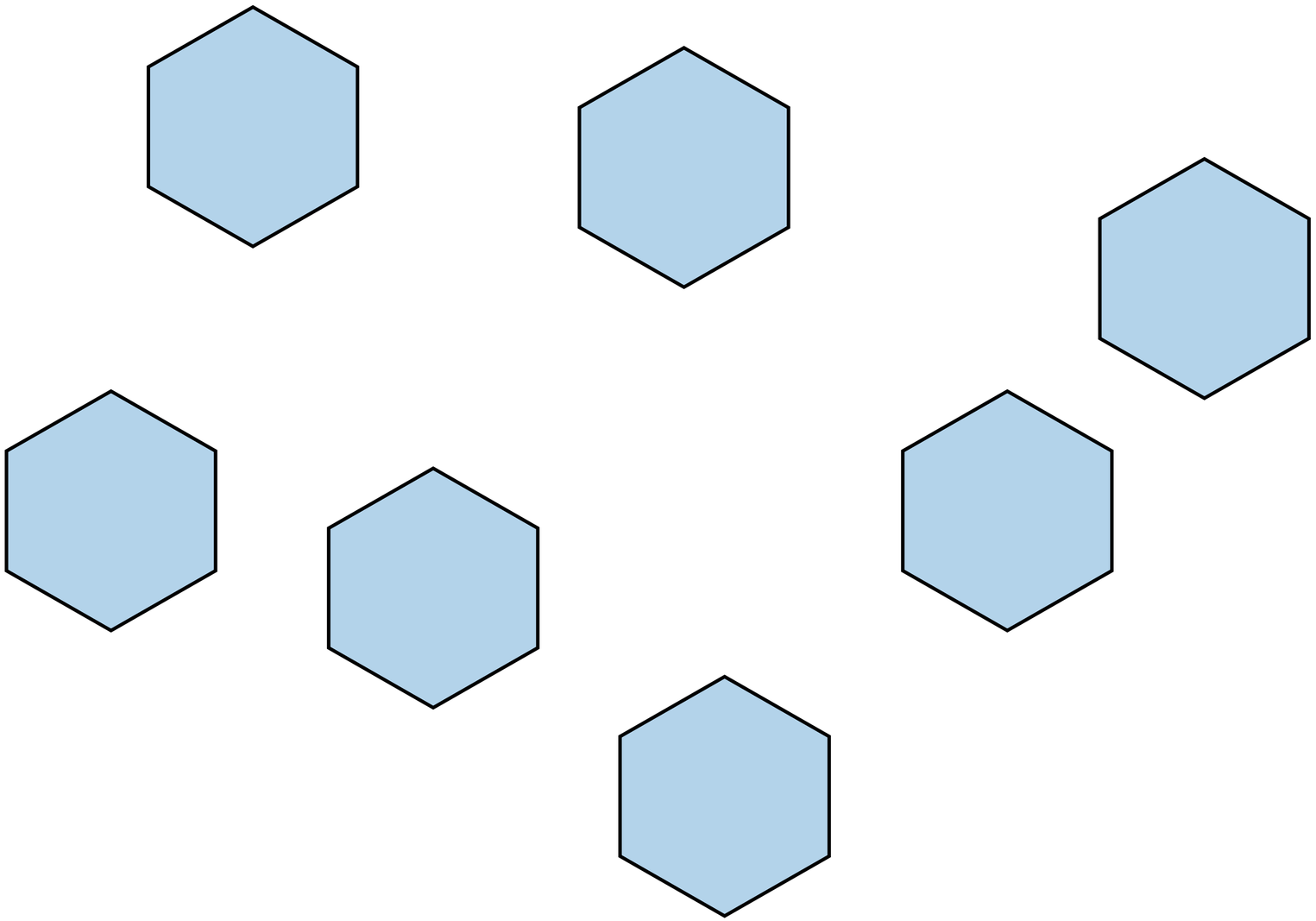
http://www.flickr.com/photos/kalexanderson/5421517469/

WHAT'S WRONG?

http://www.flickr.com/photos/kalexanderson/5421517469/

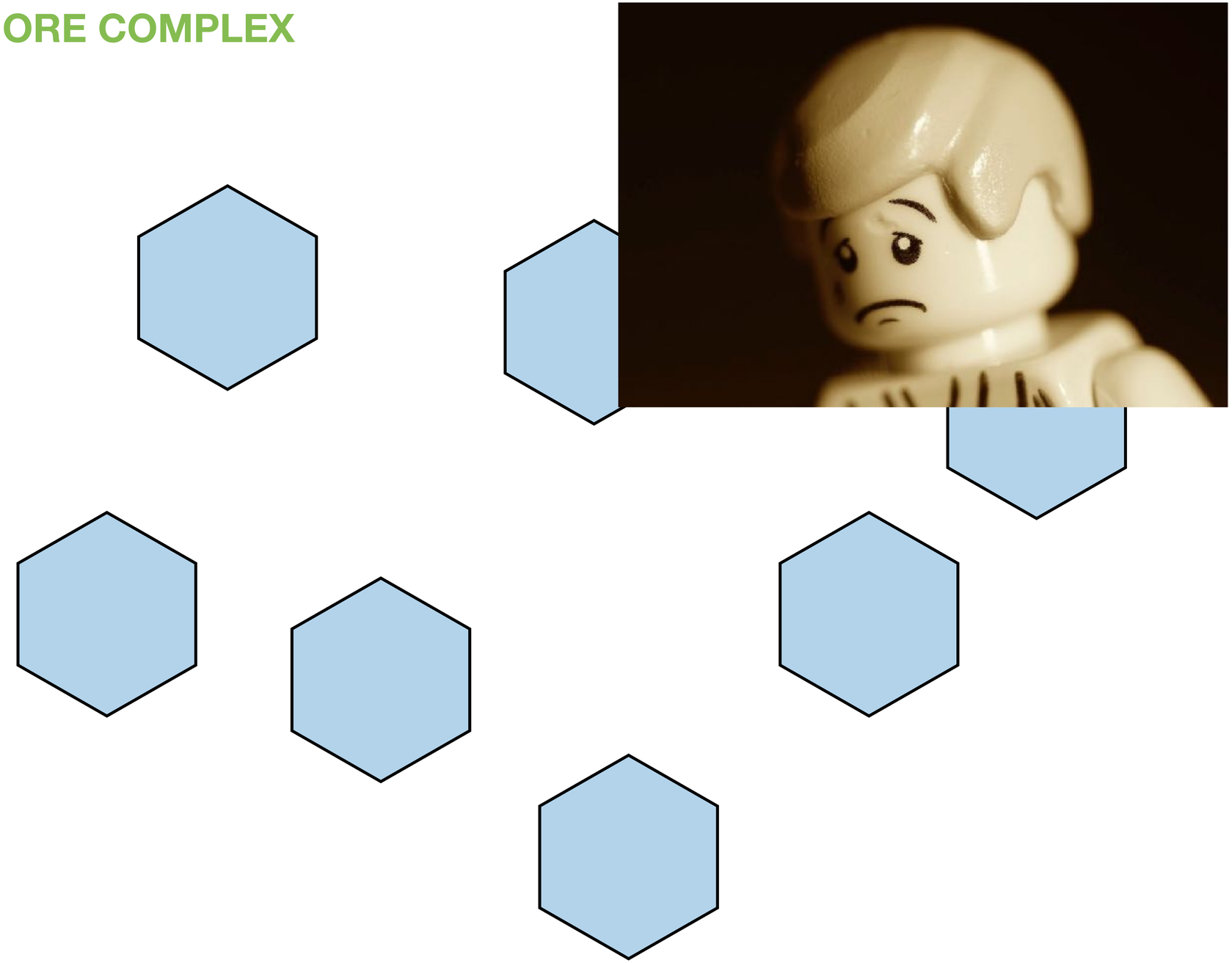
https://twitter.com/honest_update/status/651897353889259520

MONITORING IS MORE COMPLEX



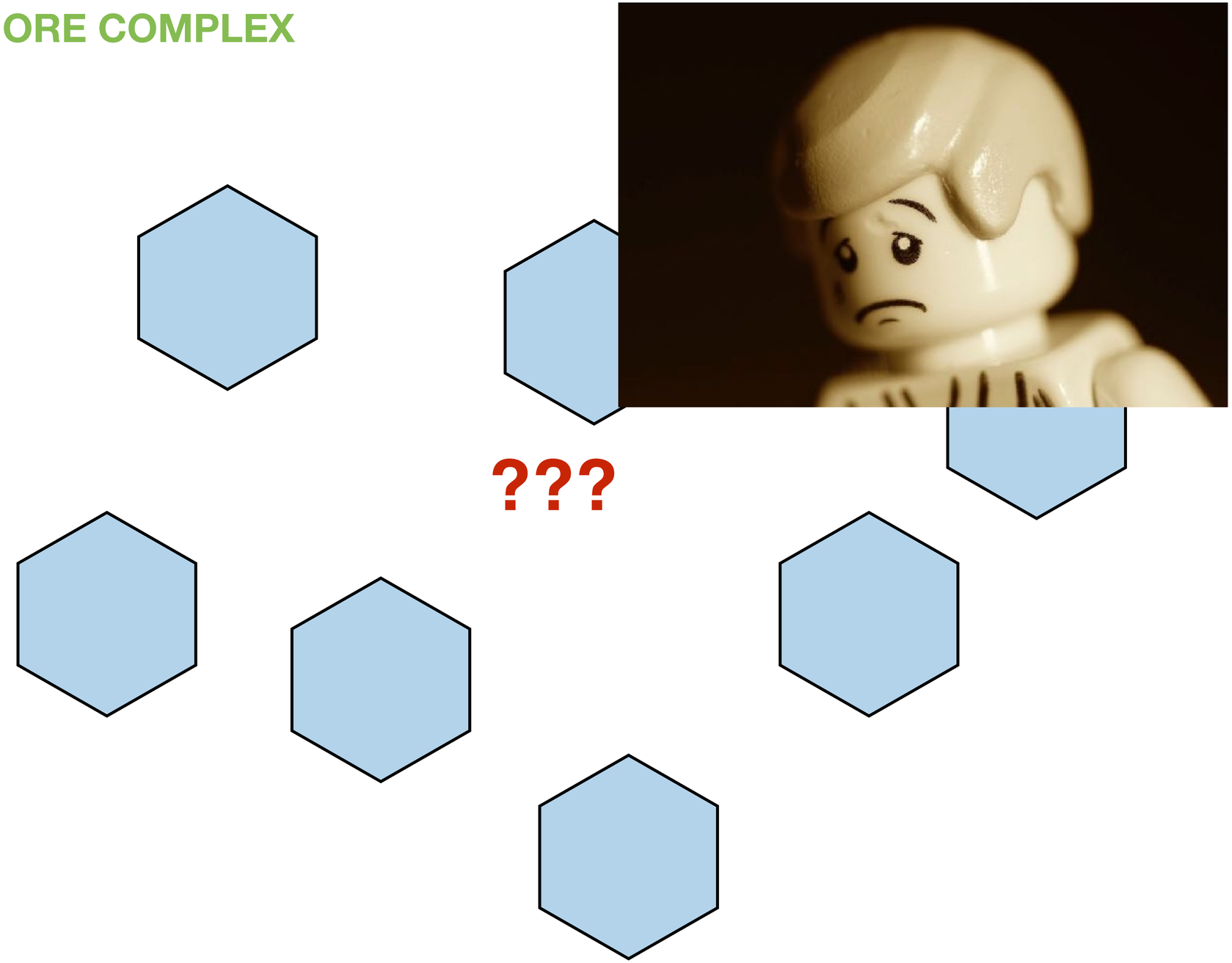
Copyright 2019 Sam Newman & Associates Ltd

MONITORING IS MORE COMPLEX



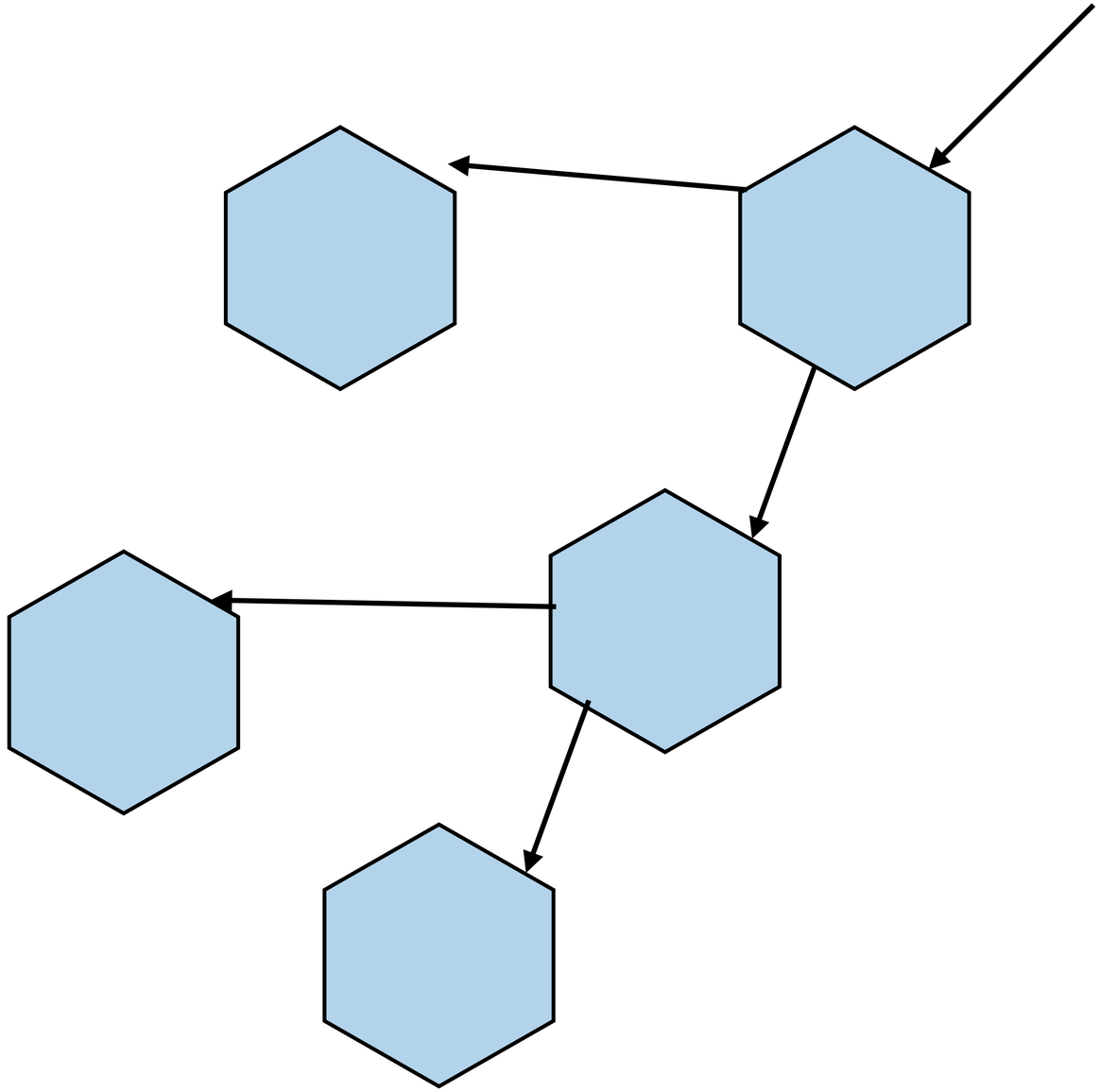
Copyright 2019 Sam Newman & Associates Ltd

MONITORING IS MORE COMPLEX



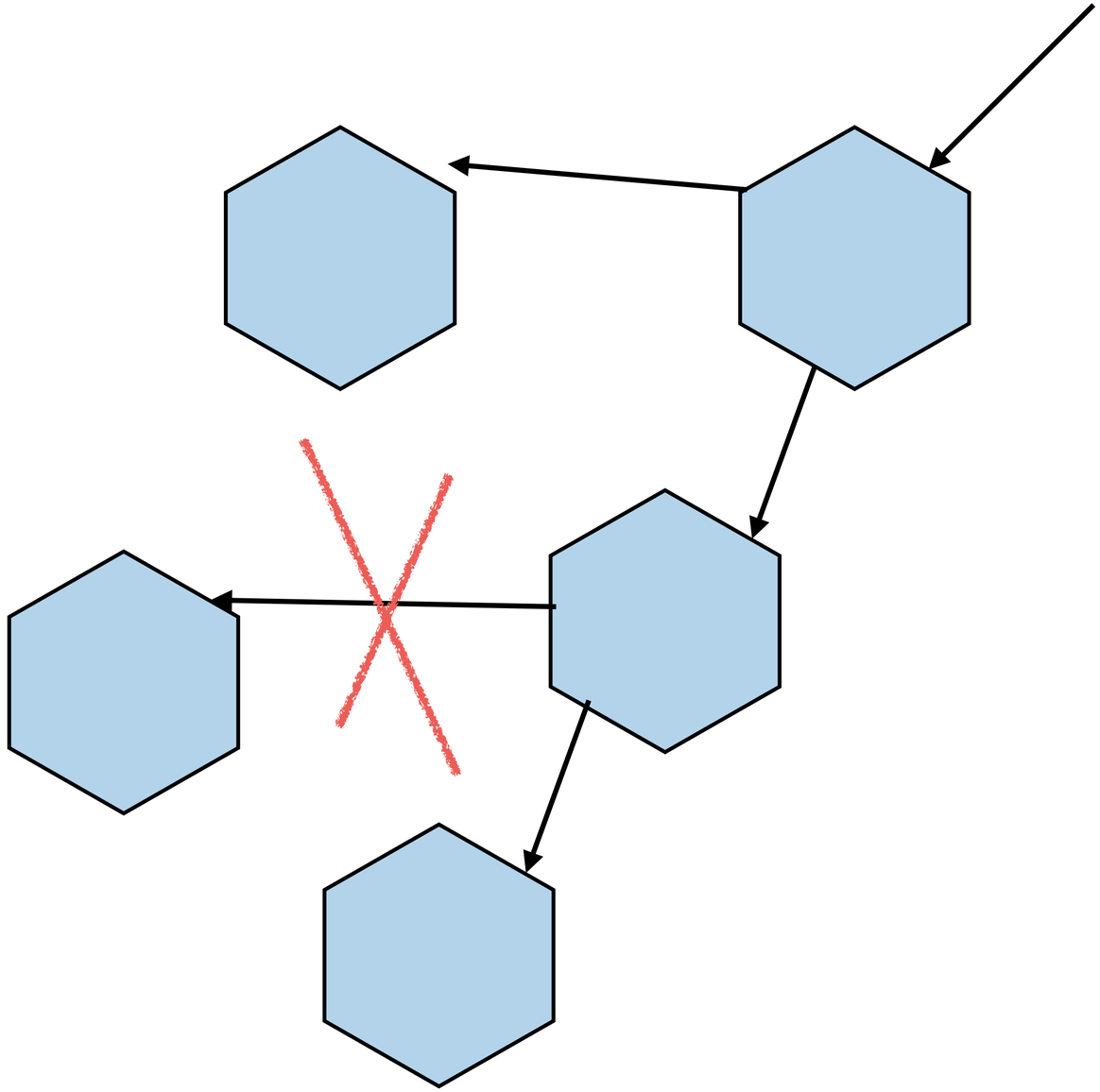
Copyright 2019 Sam Newman & Associates Ltd

RESILIENCY ISN’T FREE



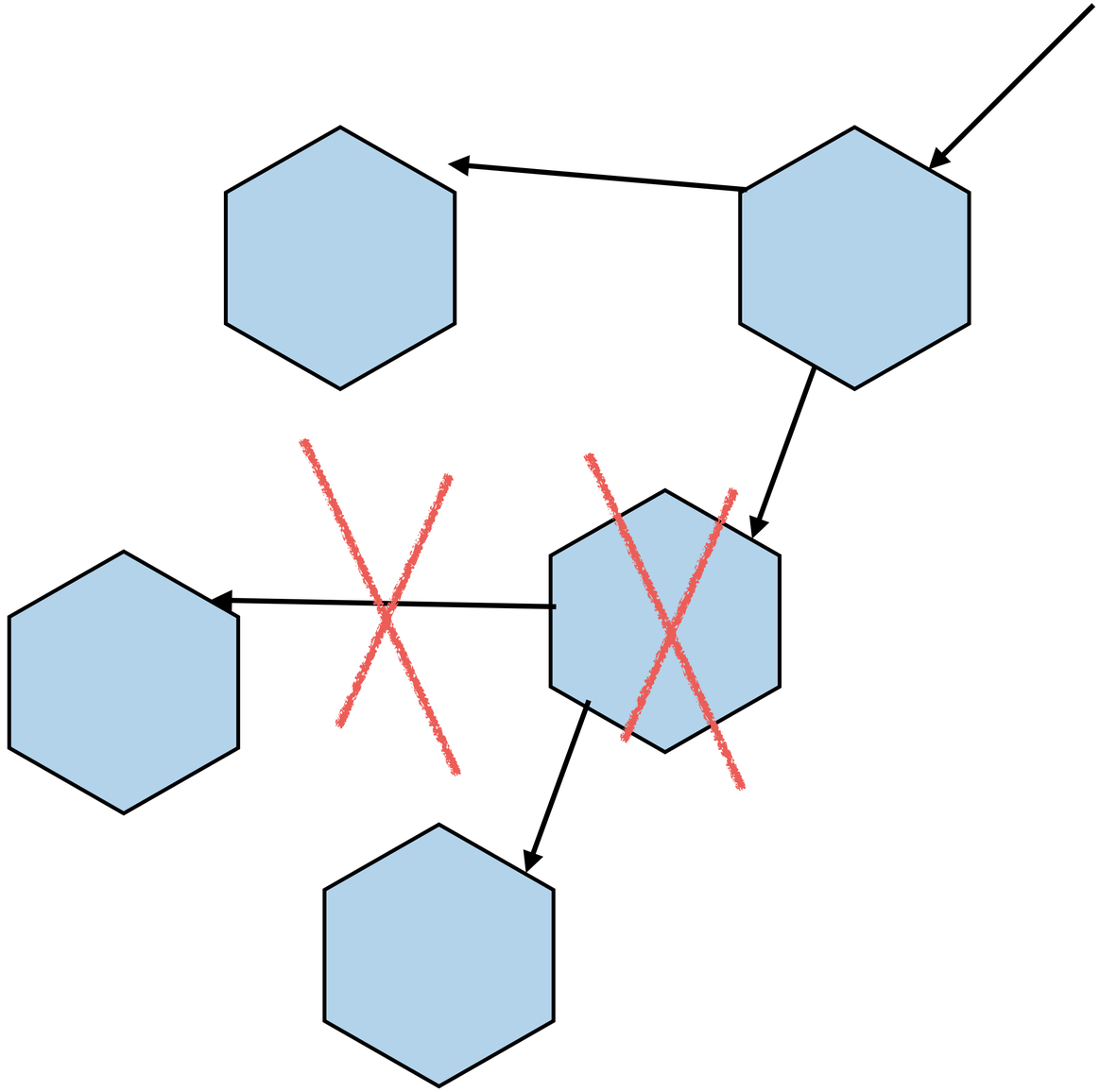
Copyright 2019 Sam Newman & Associates Ltd

RESILIENCY ISN’T FREE



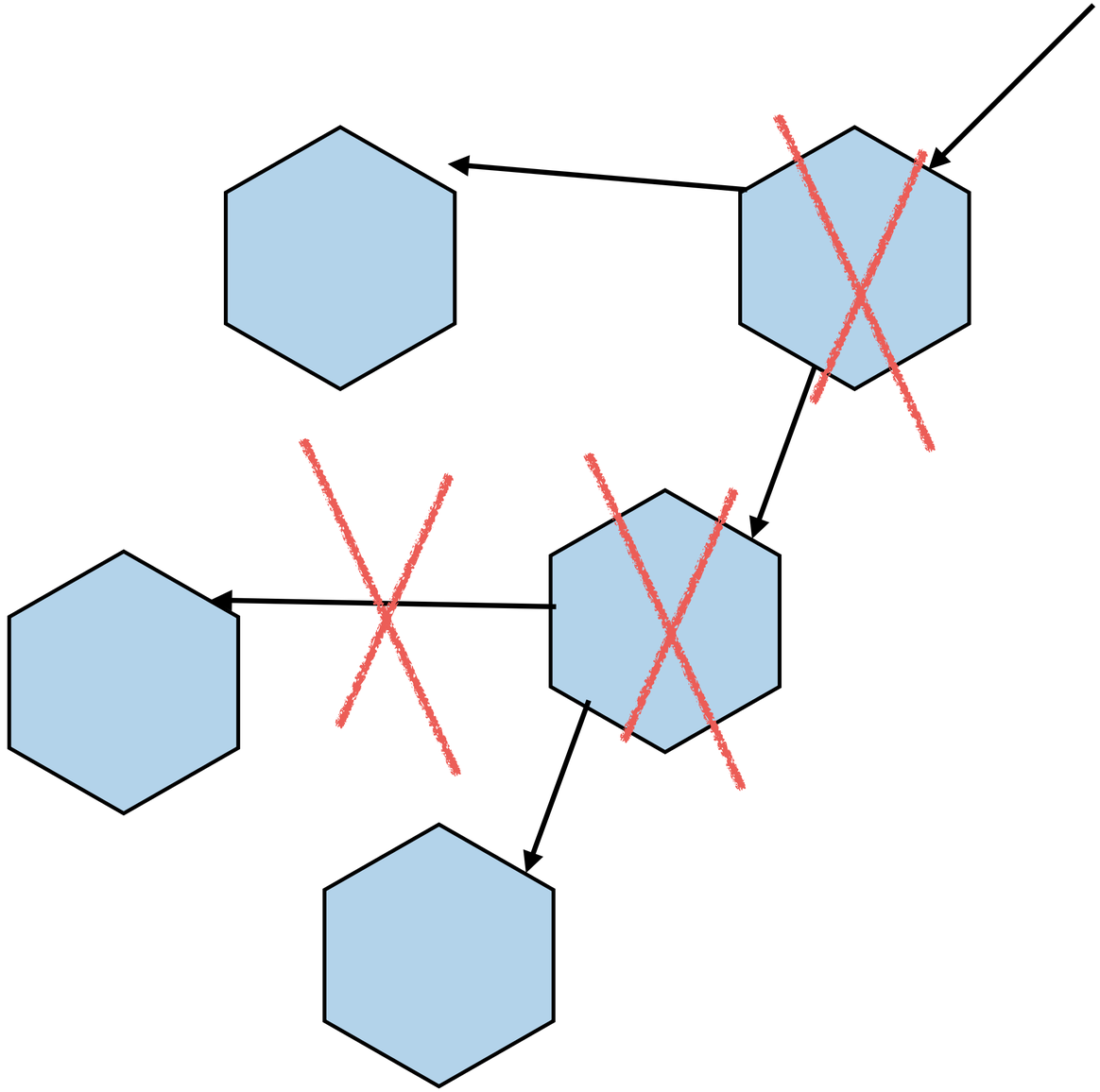
Copyright 2019 Sam Newman & Associates Ltd

RESILIENCY ISN’T FREE



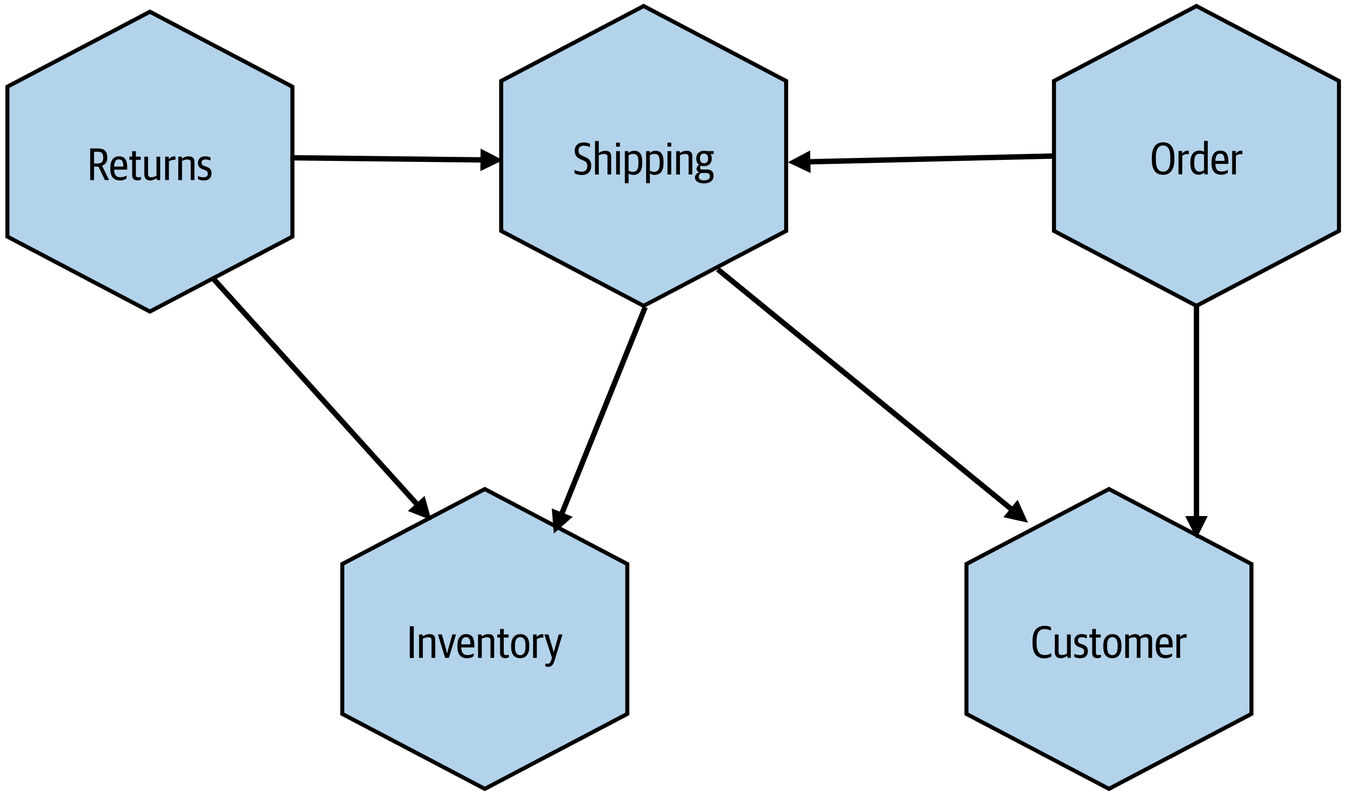
Copyright 2019 Sam Newman & Associates Ltd

RESILIENCY ISN’T FREE



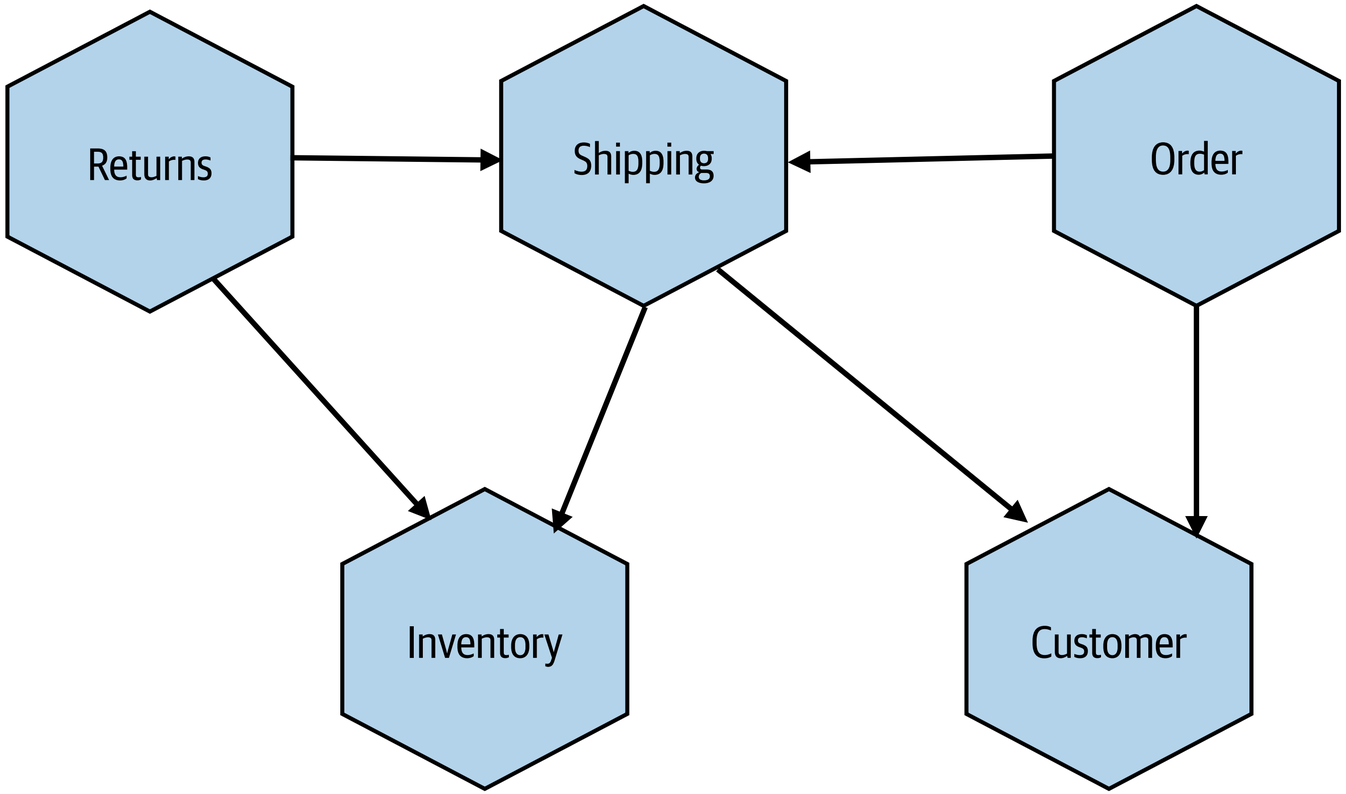
Copyright 2019 Sam Newman & Associates Ltd

SECURITY?

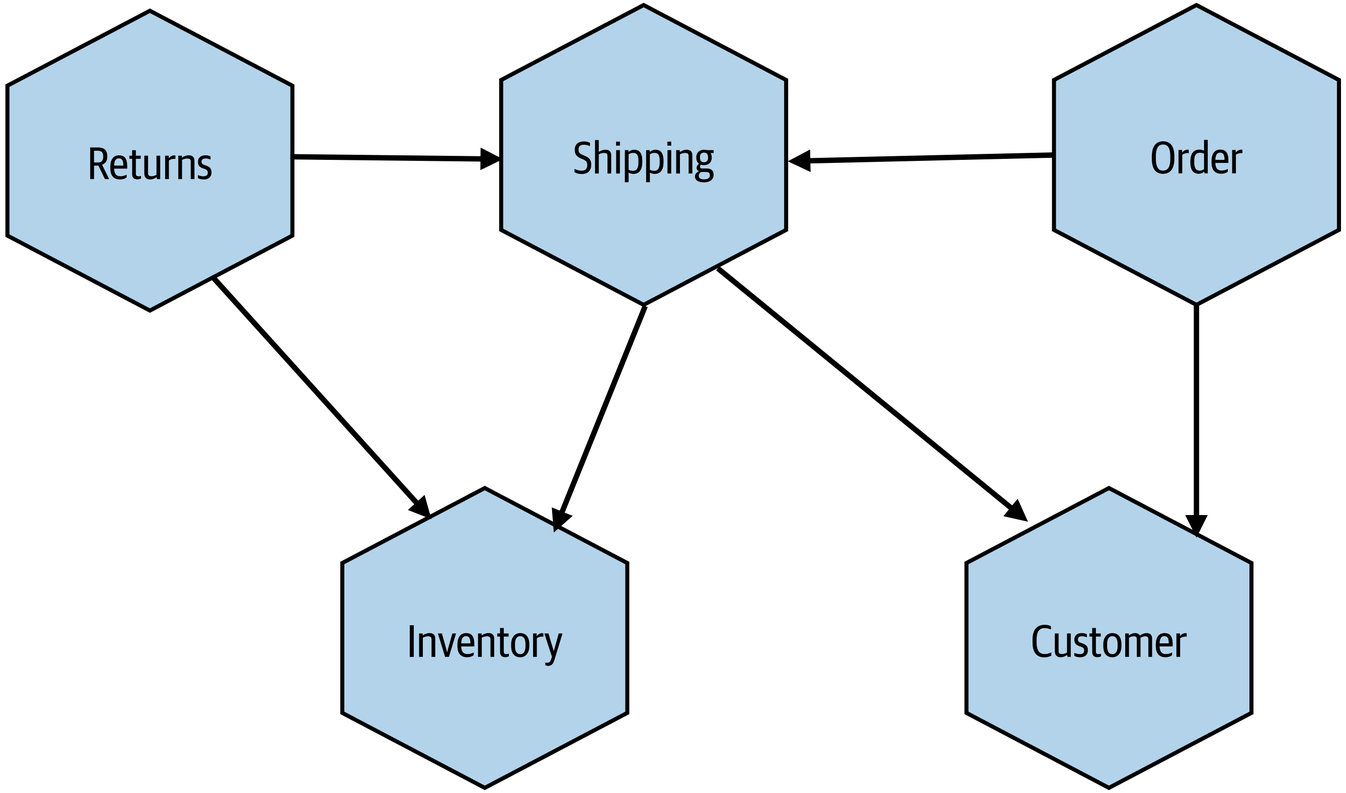


SECURITY?

Larger surface area of attack



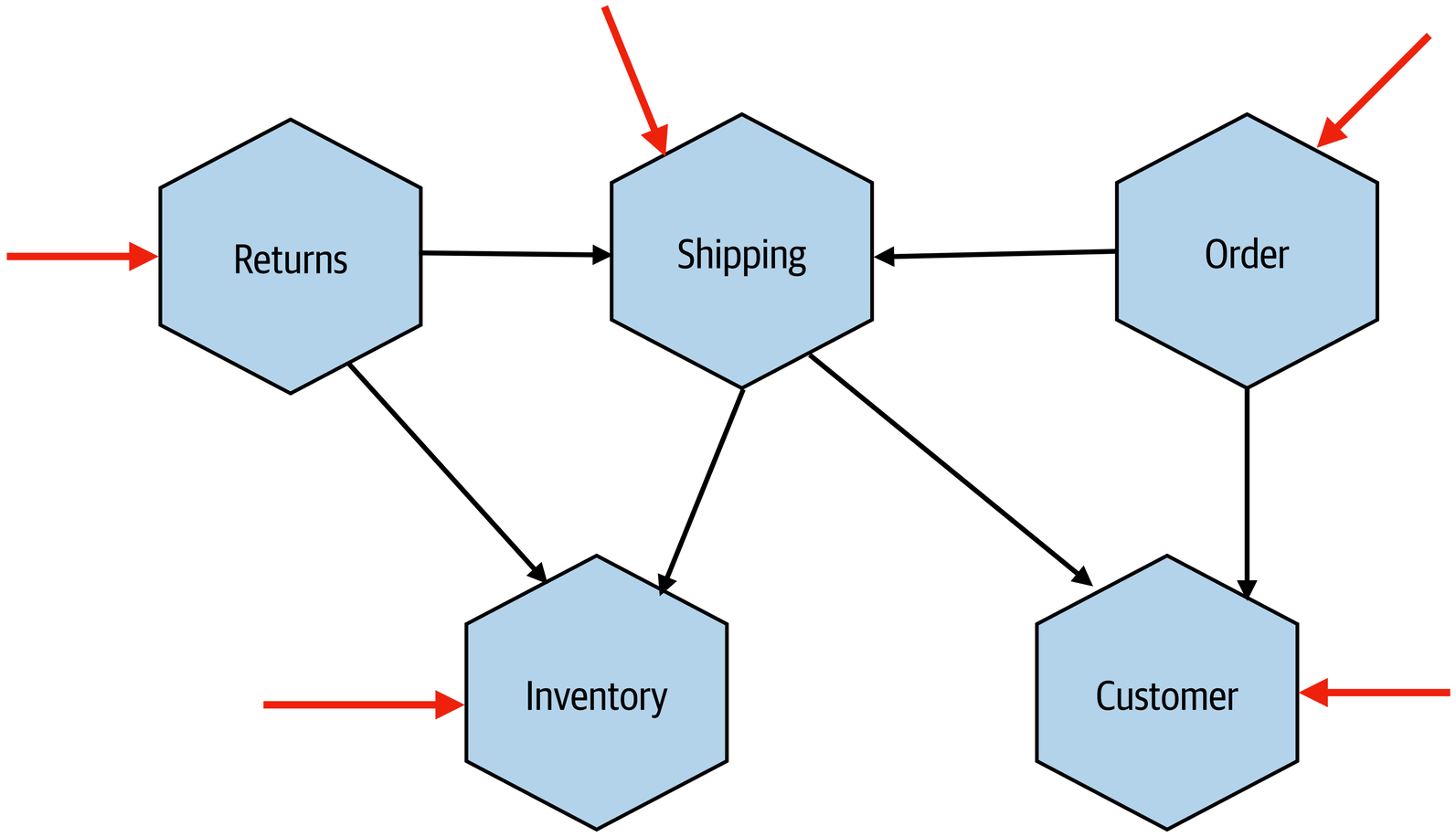
SECURITY?



Larger surface area of attack

More things to protect (patch, monitor…)

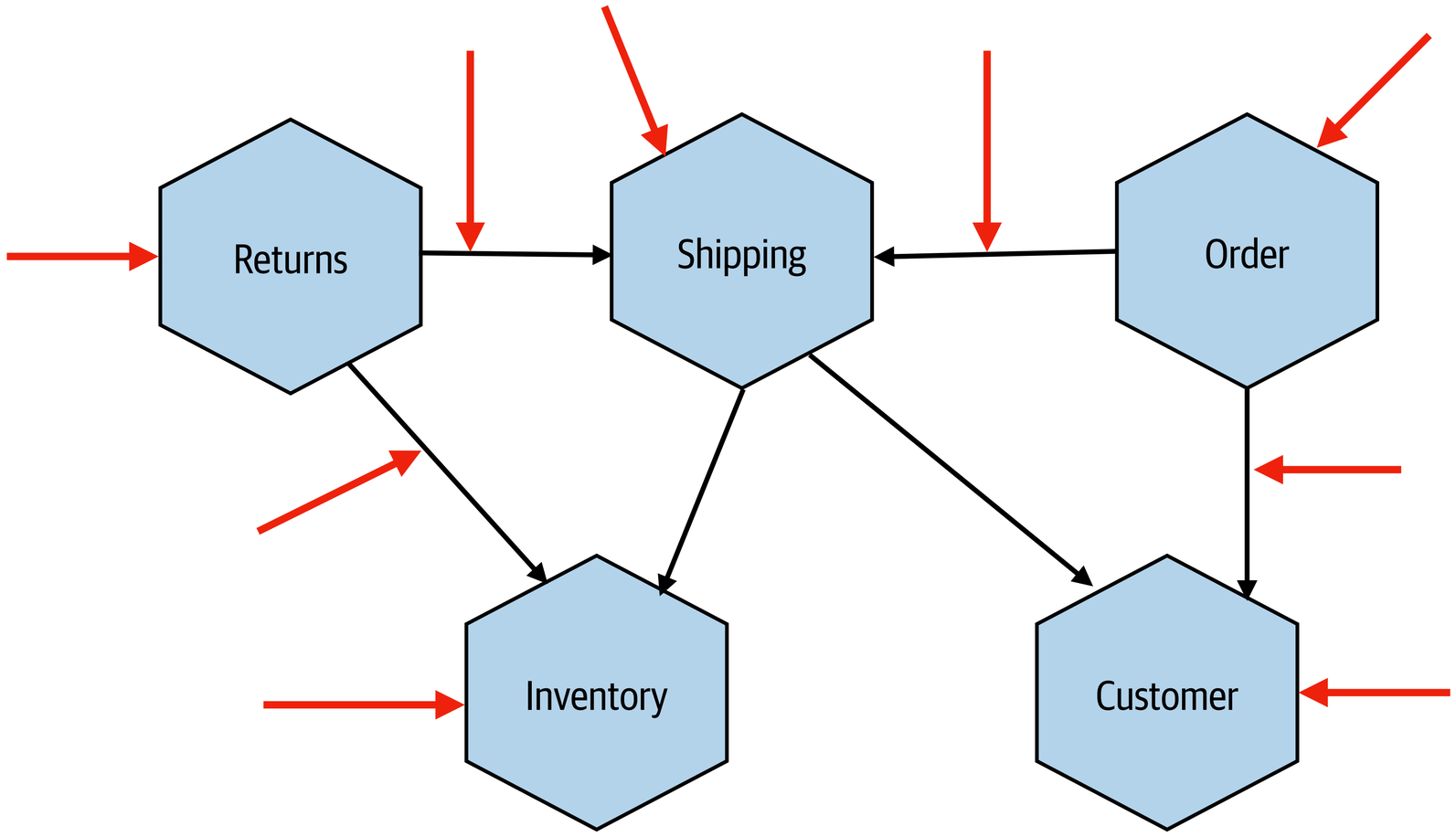
SECURITY?



Larger surface area of attack

More things to protect (patch, monitor…)

SECURITY?



Larger surface area of attack

More things to protect (patch, monitor…)

https://www.flickr.com/photos/torkildr/3462607995/

Distributed transactions

AND THE FUNDAMENTALS…

Distributed transactions

Eventual consistency

AND THE FUNDAMENTALS…

Distributed transactions

Eventual consistency

CAP theory…

AND IN GENERAL, AVOID MICROSERVICES IF…

Early-stage startups

Domain is brand new to you

Customer-installed and managed software

HOMEWORK!

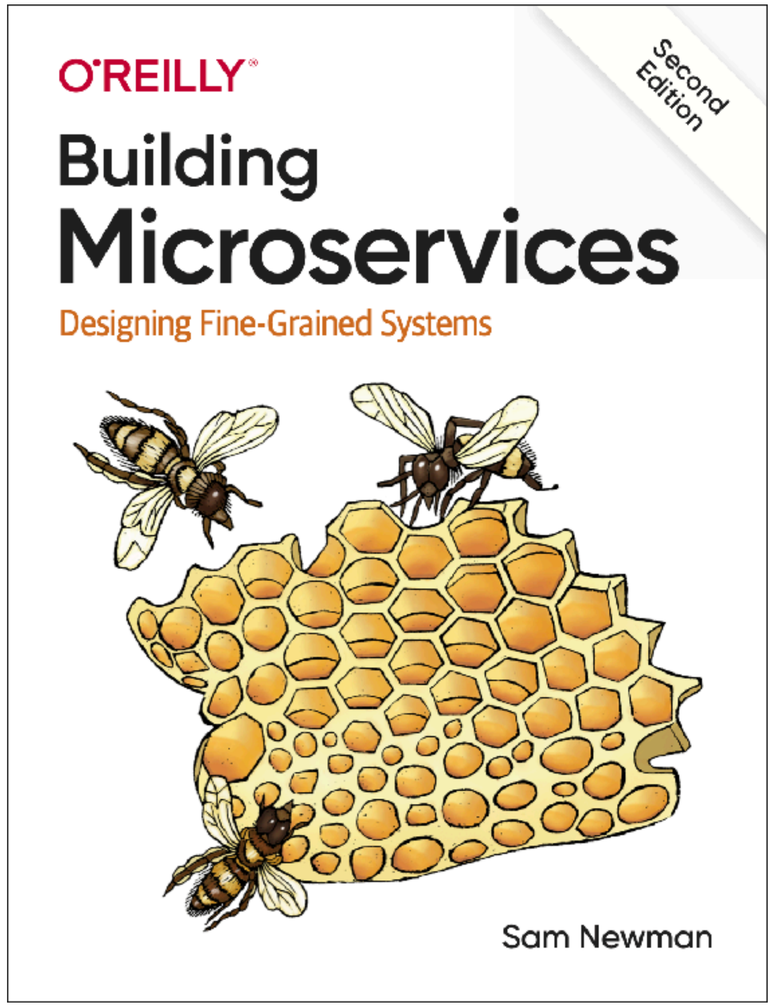
Explain why you are currently using (or thinking of using) microservices

HOMEWORK!

Explain why you are currently using (or thinking of using) microservices

Capture why this is better than the architecture you have now (or the one you moved away from)

FURTHER READING



Read chapter 1 and 2 of Building Microservices, 2nd Edition

Read What Is Domain-Driven Design

https://learning.oreilly.com/library/view/what-is-domain-driven/9781492057802/

NEXT WEEK…

Deeper dive into DDD

NEXT WEEK…

Deeper dive into DDD

Where to start?

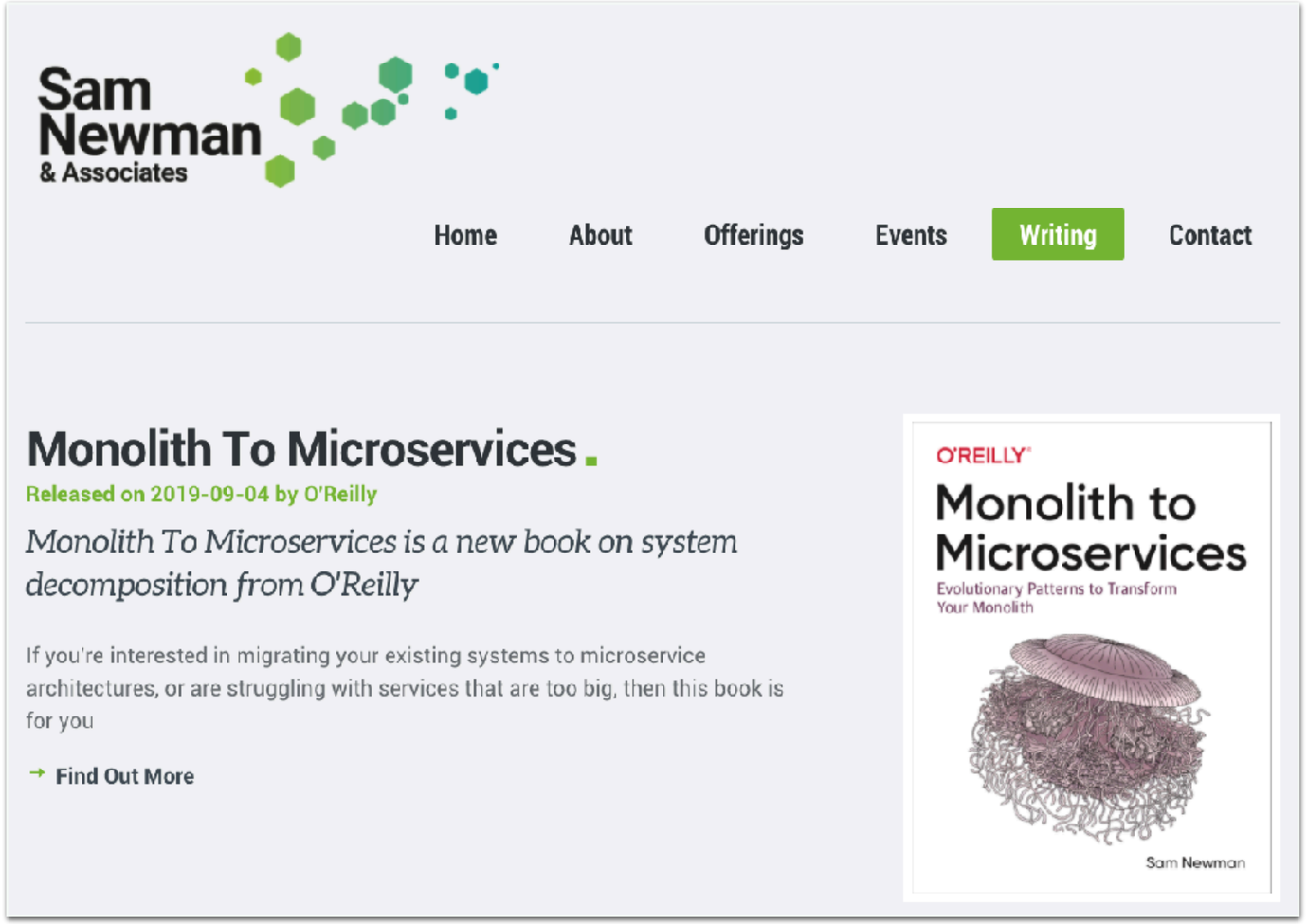
NEXT WEEK…

Deeper dive into DDD

Where to start?

Application decomposition patterns

THANKS!



Sam Newman & Associates

https://samnewman.io/